

Cribl Edge Documentation Manual

Version: v4.17

Generated: 2026-04-22 09:03:47

Copyright (c) 2017- 2026 Cribl, Inc. All rights reserved.

1. About Cribl Edge 14
1.1. Basic Concepts 16
1.2. Log in to Cribl 20
1.3. QuickConnect 22
1.4. Onboard Faster With Packs 28
1.4.1. Get Pre-Built Pack Content 29
2. Explore Cribl Edge 34
2.1. Explore Cribl Edge Nodes 37
3. Cribl Edge on Linux 47
3.1. Install Cribl Edge on Linux 48
3.2. Enable Start on Boot on Linux 55
3.3. OS Tuning for Large Deployments 61
3.4. Run Edge as an Unprivileged Linux User 68
3.5. Install Edge on Linux via RPM 70
3.6. Install Edge on Linux via .deb 74
3.7. FIPS Mode for Cribl Edge (Linux) 77
3.8. Use ACLs to Allow Cribl Edge to Read Files 84
4. Cribl Edge on Windows 87
4.1. Install Cribl Edge on Windows 89
4.2. Windows User Permissions and Requirements 105
4.3. System Proxy Configuration on Windows 110
4.4. Windows Troubleshooting 114
4.5. FIPS Mode for Cribl Edge (Windows) 116
5. Cribl Edge on MacOS 122
5.1. Install Cribl Edge on MacOS 124
5.2. Deploy Cribl Edge on MacOS with JAMF 127
6. Cribl Edge in Containers 129
6.1. Run Cribl Edge in a Docker Container 130
6.2. Deploy Cribl Edge via Kubernetes 135
7. Set Up Cribl Edge 145
7.1. Get Started with Cribl Edge 146
7.2. Cribl.Cloud vs. Self-Hosted 150
7.3. Learn About Connected Environments 154
7.3.1. How to Connect On-Prem Leaders to Cribl.Cloud 159
7.3.2. Simplify Billing with Universal Subscription 163
7.3.3. Data Payloads for Connected Environments 165

7.3.4. Send Data from On-Prem to Cribl.Cloud 170
7.4. Tutorials 173
7.4.1. Getting Started Tutorial 174
7.4.2. Configure Cribl Edge 181
7.4.3. Forward Custom Data with Cribl Edge Tags 190
7.4.4. Monitor your Infrastructure with Cribl Edge 195
7.4.5. Monitor and Troubleshoot Cribl Edge 200
7.4.6. Migrate from Third-Party Agents to Cribl Edge 205
7.5. Deployment Planning 214
7.5.1. OS and System Requirements 218
7.5.2. Plan Large-Scale Deployments of Cribl Edge 221
7.6. Set Up Leader and Edge Nodes 224
7.7. Connect Nodes to Leader Through Cribl Outpost 230
7.7.1. Set Up Cribl Outpost 233
7.7.2. Manage Cribl Outpost 236
7.8. Manage Edge Node Config Bundles 240
7.9. Leader High Availability/Failover 244
7.9.1. High Availability Requirements 246
7.9.2. Configure Standby Leader Nodes 250
7.9.3. Manage High Availability Deployment 253
7.10. Anti-Virus Exceptions 255
7.11. Ports 257
7.12. Configure Environment Variables 262
7.13. About Cribl.Cloud 266
7.13.1. Register Cribl.Cloud Organization 268
7.13.2. Manage Cribl.Cloud Organization 270
7.13.3. Cribl.Cloud Enterprise 272
8. Manage Fleets of Edge Nodes 276
8.1. Fleets 279
8.1.1. Design Fleet Hierarchy 284
8.1.2. Fleet Inheritance 288
8.1.3. Configure Fleet Settings 290
8.1.4. Create and Manage Fleets and Subfleets 301
8.1.5. Configure Fleets and Deploy Changes 306
8.1.6. Recover a Deleted Fleet 310
8.2. Edge Nodes 311
8.2.1. Manage Edge Nodes 313
8.2.2. Add and Update Edge Nodes 317

8.2.3. Filter Edge Nodes 321
8.2.4. Map Edge Nodes to Fleets 330
9. Workspaces 335
9.1. Configure Workspaces 340
10. Administration 343
10.1. Upgrade Cribl Edge 344
10.1.1. Upgrade Leader Node 346
10.1.2. Upgrade Outpost 351
10.1.3. Edge Node Upgrades 352
10.1.3.1. Upgrade Edge Nodes in a Fleet 354
10.1.3.2. Upgrade Edge Nodes Manually 357
10.1.4. Upgrade Containers 360
10.1.5. Better Practices for Upgrading Cribl Edge 363
10.1.6. Monitor Node Upgrade Status 366
10.1.7. Fleet Upgrading Use Case 369
10.1.8. Backup & Rollback via the UI 370
10.1.9. Manual Rollback 371
10.1.10. Troubleshoot Edge Upgrades 373
10.2. Notifications 376
10.2.1. Email Notifications 383
10.2.2. Notification Targets 388
10.2.2.1. Webhook Notification Targets 390
10.2.2.2. PagerDuty Notification Targets 397
10.2.2.3. Slack Notification Targets 399
10.2.2.4. AWS SNS Notification Targets 401
10.2.2.5. Email Notification Targets 405
10.3. Uninstall Cribl Edge from Linux 408
10.4. Uninstall Cribl Edge from Windows 409
10.5. Uninstall Cribl Edge from MacOS 410
10.6. Customize the Interface 411
10.6.1. Custom Banners 412
10.6.2. Custom Login Page 413
10.7. Scripts 414
11. Secure Your Deployment 416
11.1. Secure Your Cribl.Cloud Deployment 417
11.2. Secure Your On-Prem/Hybrid Deployment 420
11.3. Secure Leader/Edge Nodes Communication 427
11.3.1. How to Secure the Auth Token for the Leader Node 432

11.3.2. TLS Defaults and System-wide Settings 435
11.3.3. Secure Cribl.Cloud with TLS and Mutual TLS 440
11.3.4. Configure mTLS Authentication On-Prem 444
11.3.5. Import Certificates and Keys 448
11.3.6. Configure TLS for API and UI Access 451
11.3.7. Secure Sources and Destinations with Certificates 454
11.3.8. Configure System Proxy 458
11.3.9. Configure SOCKS Proxy 463
11.4. Manage Secrets and Keys 468
11.4.1. Create and Manage Encryption Keys 469
11.4.2. Create and Manage Secrets in Cribl Edge 471
11.4.3. Configure KMS Providers 477
11.5. Compliance 485
11.5.1. FIPS Mode for Cribl Edge (Linux) 486
11.5.2. FIPS Mode for Cribl Edge (Windows) 493
11.5.3. SELinux (Enforcing Mode) Configuration 499
12. Integrations 505
12.1. Common Data Flows 506
12.1.1. Collect Logs with File Monitor 507
12.1.2. Collect Windows Event Logs and Metrics 509
12.1.3. Collect Kubernetes Logs, Events, and Metrics 513
12.1.4. Cribl Edge to Cribl Stream 517
12.1.5. Send Data to Loki and Prometheus 521
12.2. Sources Overview 523
12.2.1. System 530
12.2.1.1. Exec Source 531
12.2.1.2. File Monitor Source 538
12.2.1.3. Journal Files Source 550
12.2.1.4. Kubernetes Events Source 554
12.2.1.5. Kubernetes Logs Source 560
12.2.1.6. Kubernetes Metrics Source 568
12.2.1.7. System Metrics Source 573
12.2.1.8. System State Source 580
12.2.2. Internal 593
12.2.2.1. Cribl HTTP Source 594
12.2.2.2. Cribl TCP Source 603
12.2.2.3. Cribl Internal Source 610
12.2.2.4. Datagen Source 615

12.2.3. Amazon 618
12.2.3.1. Amazon Data Firehose Source 619
12.2.4. Prometheus 628
12.2.4.1. Prometheus Edge Scraper Source 629
12.2.4.2. Prometheus Remote Write Source 637
12.2.4.3. Grafana Source 646
12.2.4.4. Loki Source 655
12.2.5. Splunk 664
12.2.5.1. Splunk HEC Source 665
12.2.5.2. Splunk TCP Source 681
12.2.6. Windows 691
12.2.6.1. Windows Event Forwarder Source 692
12.2.6.1.1. Client Certificate Authentication for Windows Event Forwarder 709
12.2.6.1.2. Kerberos Authentication for Windows Event Forwarder 713
12.2.6.2. Windows Event Logs Source 718
12.2.6.3. Windows Metrics Source 724
12.2.7. AppScope Source 731
12.2.8. Cloudflare Source 732
12.2.9. Datadog Agent Source 744
12.2.10. Elasticsearch API Source 756
12.2.11. HTTP/S (Bulk API) Source 768
12.2.12. Raw HTTP/S Source 780
12.2.13. Metrics Source 790
12.2.14. Model Driven Telemetry Source 798
12.2.15. OpenTelemetry (OTel) Source 804
12.2.16. SNMP Trap Source 814
12.2.17. Syslog Source 821
12.2.18. TCP JSON Source 834
12.2.19. TCP (Raw) Source 842
12.2.20. UDP (Raw) Source 850
12.2.21. Zscaler Cloud NSS Source 858
12.3. Destinations 870
12.3.1. Manage Destinations 878
12.3.2. Internal 883
12.3.2.1. Cribl HTTP Destination 884
12.3.2.2. Cribl TCP Destination 895
12.3.2.3. Cribl Search Destination 904
12.3.2.4. Disk Spool Destination 912

12.3.2.5. Default Destination 915
12.3.2.6. DevNull Destination 916
12.3.2.7. Output Router Destination 917
12.3.3. Amazon 921
12.3.3.1. Amazon CloudWatch Logs Destination 922
12.3.3.2. Amazon Kinesis Data Streams Destination 929
12.3.3.3. Amazon S3 Compatible Stores Destination 938
12.3.3.4. Amazon SQS Destination 949
12.3.3.5. Amazon Security Lake Destination 956
12.3.4. Azure 964
12.3.4.1. Azure Blob Storage Destination 965
12.3.4.2. Azure Data Explorer Destination 973
12.3.4.3. Azure Event Hubs Destination 987
12.3.4.4. Azure Monitor Logs Destination 996
12.3.4.5. Microsoft Sentinel Destination 1005
12.3.5. Elastic 1017
12.3.5.1. Elasticsearch Destination 1018
12.3.5.2. Elastic Cloud Destination 1028
12.3.6. Google Cloud 1036
12.3.6.1. Google Security Operations (SecOps) Destination 1037
12.3.6.2. Google Cloud Chronicle API Destination 1048
12.3.6.3. Google Cloud Logging Destination 1057
12.3.6.4. Google Cloud Pub/Sub Destination 1064
12.3.6.5. Google Cloud Storage Destination 1071
12.3.7. Kafka 1078
12.3.7.1. Kafka Destination 1079
12.3.7.2. Confluent Cloud Destination 1090
12.3.7.3. Amazon MSK Destination 1101
12.3.8. Metrics 1112
12.3.8.1. Graphite Destination 1113
12.3.8.2. StatsD Destination 1118
12.3.8.3. StatsD Extended Destination 1123
12.3.9. New Relic Ingest 1128
12.3.9.1. New Relic Events Destination 1129
12.3.9.2. New Relic Logs & Metrics Destination 1137
12.3.10. Prometheus 1146
12.3.10.1. Prometheus Destination 1147
12.3.10.2. Grafana Cloud Destination 1155

12.3.10.3. Loki Destination 1165
12.3.11. Splunk 1174
12.3.11.1. Splunk HEC Destination 1175
12.3.11.2. Splunk Single Instance Destination 1187
12.3.11.3. Splunk Load Balanced Destination 1195
12.3.12. CrowdStrike Falcon LogScale Destination 1208
12.3.13. ClickHouse Destination 1216
12.3.14. Cloudflare R2 Destination 1226
12.3.15. Cortex XSIAM Destination 1234
12.3.16. CrowdStrike Falcon Next-Gen SIEM Destination 1243
12.3.17. Databricks Destination 1252
12.3.18. Datadog Destination 1258
12.3.19. Dynatrace HTTP Destination 1268
12.3.20. Dynatrace OTLP Destination 1277
12.3.21. Exabeam Security Operations Platform Destination 1286
12.3.22. Microsoft Fabric Real-Time Intelligence 1292
12.3.23. Filesystem/NFS Destination 1298
12.3.24. Honeycomb Destination 1304
12.3.25. InfluxDB Destination 1312
12.3.26. MinIO Destination 1320
12.3.27. OpenTelemetry (OTel) Destination 1329
12.3.28. SentinelOne AI SIEM Destination 1339
12.3.29. SentinelOne DataSet Destination 1348
12.3.30. ServiceNow Cloud Observability 1357
12.3.31. SignalFx Destination 1367
12.3.32. SNMP Trap Destination 1375
12.3.33. Sumo Logic Destination 1378
12.3.34. Syslog Destination 1388
12.3.35. TCP JSON Destination 1402
12.3.36. Wavefront Destination 1410
12.3.37. Webhook Destination 1417
12.4. Manage Backpressure 1435
12.4.1. About Persistent Queues 1436
12.4.2. Optimize Source Persistent Queues 1442
12.4.3. Optimize Destination Persistent Queues 1454
12.4.4. About Destination Backpressure Triggers 1467
12.4.5. Backpressure Impacts to Sources 1474
12.5. About Load Balancing 1478

12.6. Transfer Data Between Workspaces or Environments 1481
12.7. Integrate with Other Services 1487
12.7.1. Cribl Edge as a Sidecar Container in AWS ECS 1488
12.7.2. Amazon Security Lake Integration 1493
12.7.3. Amazon Elastic Kubernetes Services (EKS) Add-On for Edge 1502
13. Monitor Health and Metrics 1507
13.1. Internal Metrics 1524
13.2. Internal Logs 1536
13.3. Track Data Activity for an Edge Node 1543
14. Work with Data 1549
14.1. Ingest and Inspect Data 1550
14.1.1. Onboard Data 1551
14.1.2. Use Datagens to Simulate Live Data 1554
14.1.3. Create and Share Data Samples 1559
14.2. Event Data Structure and Flow 1572
14.2.1. Event Model 1573
14.2.2. Event Processing Order 1575
14.2.3. Event Breakers 1577
14.2.4. Event Breaker Types 1586
14.2.4.1. Azure Virtual Network (VNet) Flow Event Breaker 1587
14.2.4.2. CSV Event Breaker 1591
14.2.4.3. File Header Event Breaker 1593
14.2.4.4. JSON Array Event Breaker 1596
14.2.4.5. JSON New Line Delimited Event Breaker 1606
14.2.4.6. Regex Event Breaker 1608
14.2.4.7. Timestamp Event Breaker 1611
14.3. Routes 1613
14.4. Pipelines 1620
14.4.1. Validate Pipeline Logic Using Data Preview 1628
14.5. Process Data with AI Assistance 1638
14.5.1. Create Pipelines With Cribl Copilot Editor 1639
14.5.2. Convert Data into a Custom Schema 1646
14.6. Manage Metrics and High Cardinality 1655
14.7. Build Custom Logic to Route and Process Your Data 1665
15. Functions 1671
15.1. Aggregations 1677
15.2. Aggregate Metrics 1687
15.3. Auto Timestamp 1697

15.4. CEF Serializer 1702
15.5. Chain 1704
15.6. Clone 1706
15.7. Code 1708
15.8. Comment 1718
15.9. DNS Lookup 1720
15.10. Drop 1725
15.11. Drop Dimensions 1727
15.12. Dynamic Sampling 1732
15.13. Eval 1735
15.14. Event Breaker Function 1740
15.15. Flatten 1743
15.16. Fold Keys 1746
15.17. GeoIP 1750
15.18. Grok 1753
15.19. JSON Unroll 1756
15.20. Lookup 1759
15.21. Mask 1765
15.22. Numerify 1773
15.23. OTLP Logs 1775
15.24. OTLP Metrics 1784
15.25. OTLP Traces 1801
15.26. Parser 1807
15.27. Publish Metrics 1819
15.28. Redis 1825
15.29. Regex Extract 1838
15.30. Regex Filter 1842
15.31. Rename 1843
15.32. Rollup Metrics 1846
15.33. Sampling 1848
15.34. Serialize 1850
15.35. SNMP Trap Serialize 1853
15.36. Suppress 1856
15.37. Tee 1859
15.38. Unroll 1862
15.39. XML Unroll 1864
15.40. Windows SID Lookup 1868
15.41. Prometheus Publisher (deprecated) 1870

15.42. Reverse DNS (deprecated) 1872
15.43. Trim Timestamp (deprecated) 1874
16. Packs 1877
16.1. Pack-Based Configuration Management 1892
16.1.1. Plan a Pack-Based Strategy 1896
16.1.2. Create a Pack 1904
16.1.3. Version Control Packs in External Git Repos 1912
16.2. Packs Publication Standards 1917
17. Reference 1929
17.1. Knowledge Objects 1930
17.1.1. Variables Library 1931
17.1.2. Regexes 1938
17.1.3. JSON Schemas 1942
17.1.4. Parquet Schemas 1946
17.1.5. AppScope Configs 1955
17.2. Cribl Expressions 1956
17.2.1. C.Crypto - Data Encryption and Decryption 1957
17.2.2. C.Decode and C.Encode - Encoding and Decoding 1960
17.2.3. C.Lookup - Inline Lookup Methods 1968
17.2.4. C.Mask - Data Masking Methods 1972
17.2.5. C.Net - Network Methods 1983
17.2.6. C.Text - Text Methods 1990
17.2.7. C.Time - Time Methods 1994
17.2.8. Miscellaneous Expression Methods 2001
17.3. CLI Reference 2010
17.3.1. auth 2014
17.3.2. boot-start 2019
17.3.3. cloud-workspace 2021
17.3.4. decrypt 2023
17.3.5. diag 2024
17.3.6. encrypt 2029
17.3.7. git 2030
17.3.8. help 2033
17.3.9. keys 2035
17.3.10. limits 2037
17.3.11. mode-master 2038
17.3.12. mode-single 2040
17.3.13. mode-edge 2041

17.3.14. mode-worker 2042
17.3.15. mode-managed-edge 2044
17.3.16. mode-outpost 2046
17.3.17. nc 2048
17.3.18. node 2050
17.3.19. pack 2052
17.3.20. parquet 2056
17.3.21. pipe 2060
17.3.22. pq 2062
17.3.23. reload 2066
17.3.24. restart 2067
17.3.25. start 2068
17.3.26. status 2069
17.3.27. stop 2070
17.3.28. vars 2071
17.3.29. version 2074
17.4. Environment Variables 2075
17.5. Configuration Files 2084
17.5.1. breakers.yml 2090
17.5.2. certificates.yml 2094
17.5.3. cribl.yml 2095
17.5.4. groups.yml 2106
17.5.5. inputs.yml 2109
17.5.6. instance.yml 2225
17.5.7. jobs.yml 2230
17.5.8. job-limits.yml 2258
17.5.9. leader.yml 2261
17.5.10. limits.yml 2264
17.5.11. logger.yml 2268
17.5.12. mappings.yml and fleet-mappings.yml 2269
17.5.13. messages.yml 2270
17.5.14. notifications.yml 2271
17.5.15. outpost.yml 2276
17.5.16. outputs.yml 2279
17.5.17. parsers.yml 2394
17.5.18. persistent-queue.yml 2397
17.5.19. regexes.yml 2398
17.5.20. samples.yml 2399

17.5.21. schemas.yml 2402
17.5.22. scripts.yml 2403
17.5.23. service.yml 2404
17.5.24. vars.yml 2407
17.6. Kubernetes Metrics Details 2408
17.7. Linux System Metrics Details 2436
17.8. Windows System Metrics Details 2453
18. Troubleshoot Cribl Edge 2491
18.1. Common Challenges on the Edge 2492
18.2. Diagnose Issues 2494

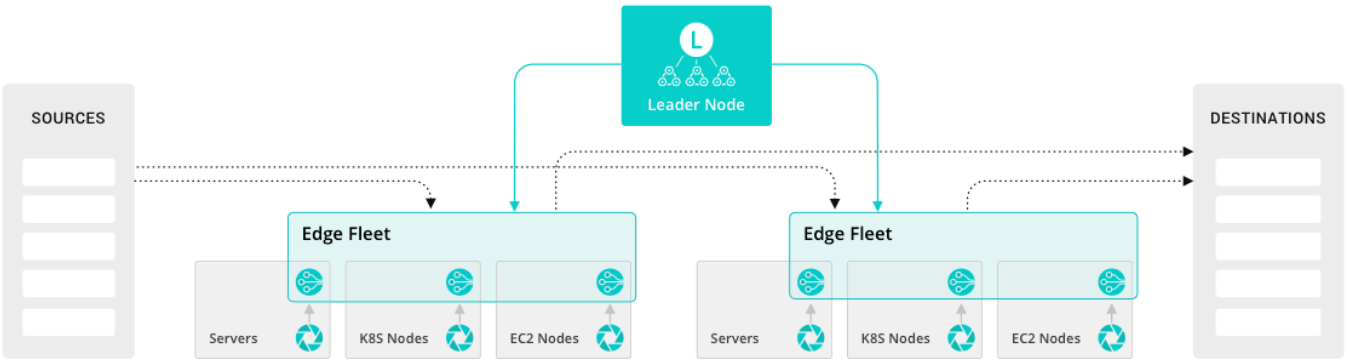
1. About Cribl Edge

What Is Cribl Edge?

Cribl Edge helps you collect and process observability data - logs, metrics, and application data - in real time, from your Linux, Windows, and macOS machines, apps, or microservices, and deliver them to Cribl Stream or any supported destination.

With Cribl Edge you can:

- Automatically discover and collect host, container, and application metrics, as well as logs, on endpoints.
- Process data with Functions and Pipelines.
- Forward data to any of the many supported destinations.
- Easily manage configurations while maintaining version control.
- Teleport to the edge for local preview and configuration validation.



Leader Node, Fleets, and Destinations

Cribl Edge is not intended for collecting data from multiple hosts. If you need to receive and aggregate data from various endpoints, use Cribl Stream instead. Cribl Stream is designed to handle high volume workloads by employing multiple Worker processes.

Conceptual Overview of Cribl Edge

This two-minute video provides a quick overview of the Cribl Edge product.

Who Is Cribl Edge For?

Cribl Edge is built for ITOps and SecOps Engineers and users of operational and security intelligence products and services who want to simplify the process of collecting and analyzing data from a host.

Cribl University offers the Cribl Certified Admin for Edge course where you will learn to implement, optimize, and manage your deployments. To follow the direct course link, first log into your Cribl University account. (To create an account, select the **Sign up** link. You'll need to click through a short **Terms & Conditions** presentation, with chill music, before proceeding to courses - but Cribl's training is always free of charge.) Once logged in, check out other useful Cribl Edge courses.

Goat-to Resources

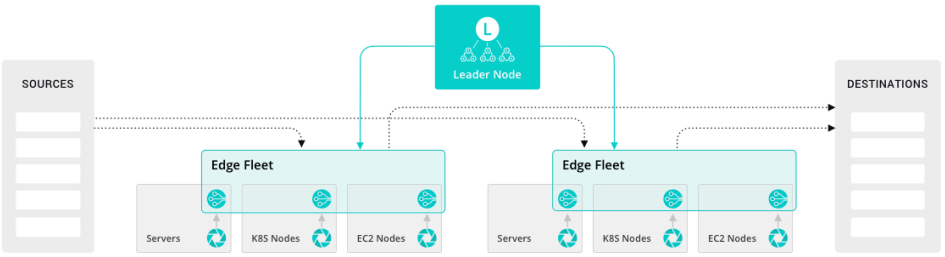
Tip Sheet: Cribl Edge — Article

1.1. Basic Concepts

Notable features and concepts to get a fundamental understanding of Cribl Edge

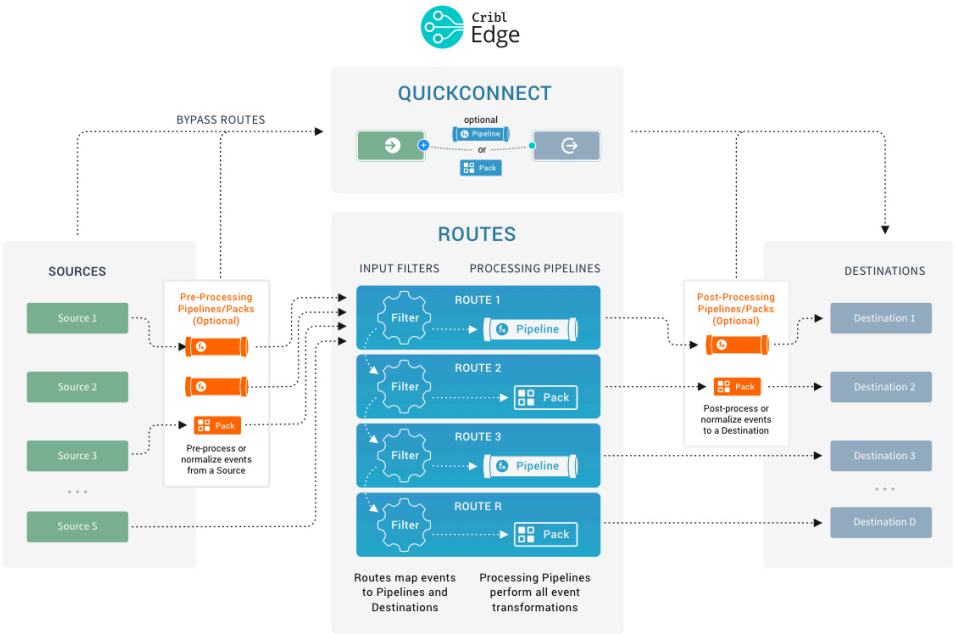
---

As we describe features and concepts, it helps to have a mental model of Cribl Edge as a system that collects and process observability data - logs, metrics, application data, etc. - in near real time, from your Linux, Windows, and macOS machines, apps, microservices etc., and delivers them to Cribl Stream or any supported destination.



Edge processing, management, and receivers

Let's zoom in on the center of the above diagram, into an Edge Node, to take a closer look at the processing and transformation options. The basic interface concepts to work with are Sources, which collect data; and Routes, which manage data flowing through Pipelines, which consist of Functions.



Routes, Pipelines, Functions

Sources

Sources are configurations that enable Edge to collect data from the local machine (logs, metrics, etc.) or to receive data from remote senders (TCP, Syslog etc.).

QuickConnect

QuickConnect is a graphical interface for setting up data flow through your Edge deployment. You can quickly drag and drop connections between Sources and Destinations, optionally including - or excluding - Pipelines or Packs.

The only major constraint is that QuickConnect completely bypasses Routes. So QuickConnect configurations have no Routing table, and no conditional cloning, cascading, or routing of data - every QuickConnect connection is parallel and independent.

To access QuickConnect, navigate to the Fleet's submenu and select **Collect**.

Routes

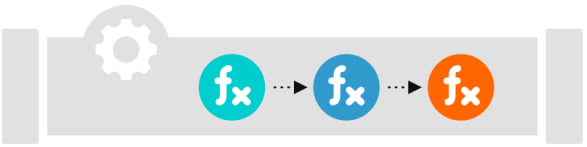
Routes evaluate incoming events against **filter** expressions to find the appropriate Pipeline to send them to. Routes are **evaluated in order**. Each Route can be associated **with only one** Pipeline and one output (configured as a Edge **Destination**).

By default, the **Final** toggle is on when each Route is created. With this setting, a Route-Pipeline-Destination set will consume events that match its filter, and that's that.

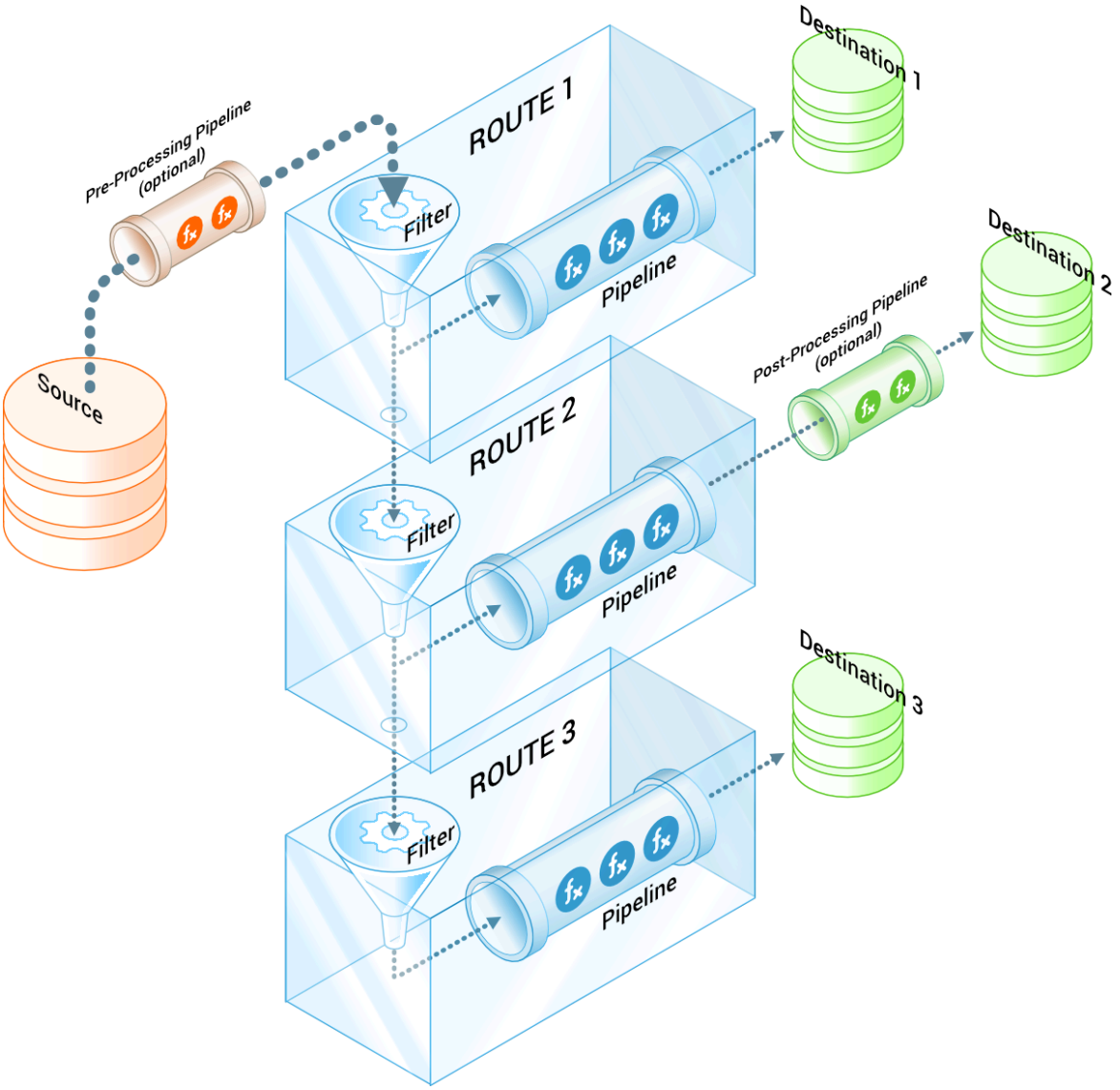
However, if you toggle **Final** off, one or more event **clones** will be sent down the associated Pipeline, while the original event continues down Edge's Routing table to be evaluated against other configured Routes. This is very useful in cases where the same set of events needs to be processed in multiple ways, and delivered to different destinations. For more details, see Routes.

Pipelines

A series of Functions is called a Pipeline, and the order in which you specify the Functions determines their execution order. Routes deliver Events to the beginning of a Pipeline, and as they're processed by a Function, the events are passed to the next Function down the line.



Pipelines attached to Routes are called **processing Pipelines**. You can optionally attach **pre-processing Pipelines** to individual Edge Sources, and attach **post-processing Pipelines** to Edge Destinations. All Pipelines are configured through the same UI - these three designations are determined only by a Pipeline's placement in Edge's data flow.

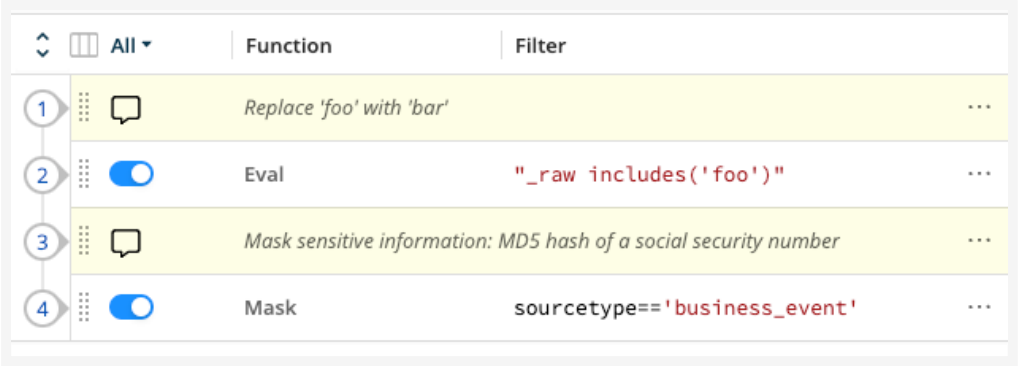


Pipelines categorized by position

Events only move forward - toward the end of a Pipeline, and eventually out of the system. For more details, see Pipelines.

Functions

At its core, a **Function** is a piece of code that executes on an event, and that encapsulates the smallest amount of processing that can happen to that event. For instance, a very simple Function can be one that replaces the term `foo` with `bar` on each event. Another one can hash or encrypt `bar`. Yet another function can add a field - say, `dc=jfk-42` - to any event with `source=*us-nyc-application.log`.



Functions stacked in a Pipeline

Functions process each event that passes through them. To help improve performance, Functions can optionally be configured with filters to limit their processing scope to matching events only. See Build Custom Logic to Route and Process Your Data: Filter Expressions for more information.

1.2. Log In To Cribl

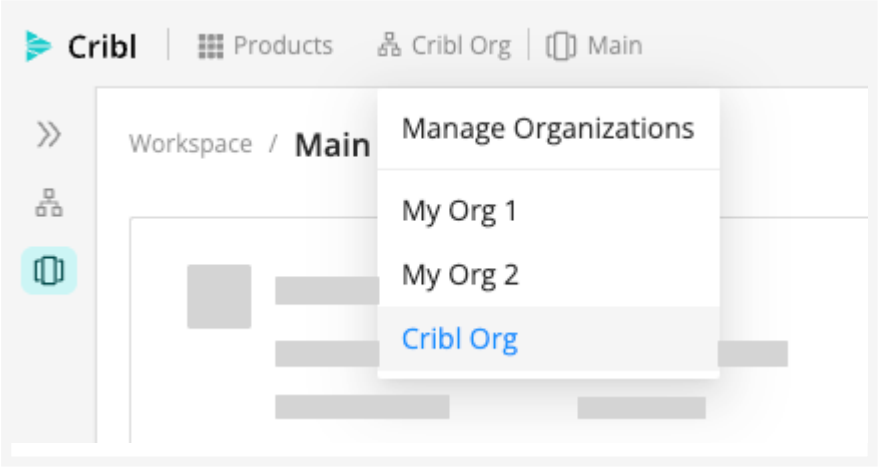
When you first log in to a Cribl instance, you will be required to configure your account and authenticate.

Depending on how your Organization is configured, you may need to either authenticate via your SSO provider (such as Okta, Entra ID, or Ping Identity), or create a password for use with your email address.

When creating a new username and password, you will be prompted for account information, such as name, company, or phone number.

Switch Between Organizations

When you are a member of multiple Organizations, you can switch between them on the top bar. Select your current Organization's name on the bar to open a drop-down with all Organizations you have access to.



Switching between Organizations

Organizations with Different Authentication Methods

If you switch between Organizations that use different authentication methods (SSO versus username/password), you will be asked to re-authenticate with the appropriate credentials.

If the main Organization uses SSO, and you are switching to an Organization that uses username/password authentication, you will need to authenticate with a password. If you are accepting an invitation to the latter Organization, you will be prompted to create a password and go through a verification process.

If the main Organization uses username/password authentication, and you are switching to an SSO-based Organization, you will need to authenticate via SSO.

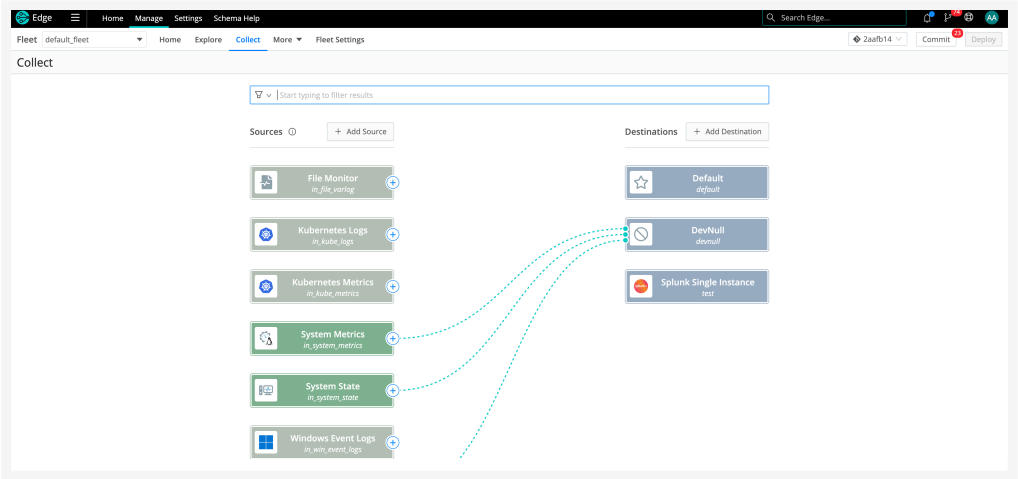
1.3. QuickConnect

Use QuickConnect to sent data from Sources to Destinations with drag-and-drop.

The QuickConnect rapid-development UI enables you to visually connect Cribl Edge inputs (Sources) to outputs (Destinations) through simple drag-and-drop.

You can insert Pipelines or Packs into the connections, to take advantage of the full range of data-transformation Functions in Cribl Edge. Or you can omit these processing stages entirely, to send incoming data directly to Destinations - with minimal configuration fuss.

QuickConnect was designed as a simple, quick way to prototype, test, and send data flowing through Cribl Edge. But it's also entirely suitable for configuring a production deployment, if your needs are restricted to sending data through parallel paths, and you don't need to use Routes. (See QuickConnect Versus Routes to help you clarify that choice.)



QuickConnect UI

You also have the option to start with simple direct connections, and later add Pipelines' and/or Packs' processing power at up to three stages of each connection.

Initiating QuickConnect

To access QuickConnect, from a single-instance or individual **Edge Node**, select **Collect**.

To display QuickConnect in a distributed deployment, you must first select a **Fleet**. Select the **Manage** link to select the `default` or a different Fleet.

You can switch to Cribl Edge's Routes UI at any time from the top nav by selecting **Routing** > **Data Routes**. To toggle back, select **Routing** > **QuickConnect**.

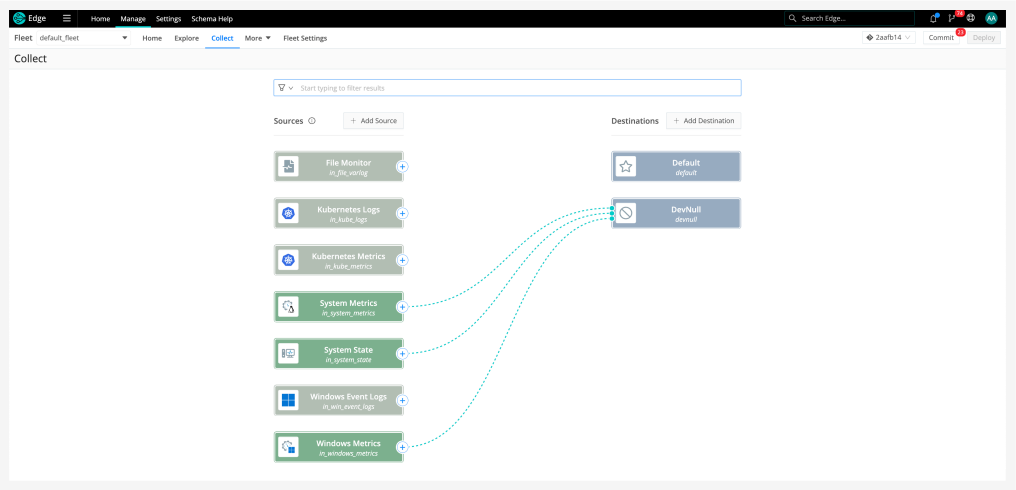
QuickConnect Versus Routes

QuickConnect provides access to most of Cribl Edge's configuration options, with a few restrictions (in exchange for the simplified visual interface):

- QuickConnect completely bypasses Routes - which is why every QuickConnect path is independent and parallel. There is no Route-level data filtering, and you forego the Cribl Edge Routing table options to clone and cascade data across Pipelines.
- Most Sources are available to fully configure, with a few current exclusions.
- Each supported Source's config can be created in either QuickConnect or Routes, and will reside only in that context. But you can switch a config between the two contexts, or you can create an identical config on the opposite side.
- All Destinations are available to fully configure.
- Everything you configure in QuickConnect exists separately from everything you configure in Data Routes. (But remember that you can use the **Routing** menu to toggle between contexts, and then replicate a QuickConnect configuration on the Data Routes side, or vice versa.)

Using QuickConnect

QuickConnect is designed to be nearly self-documenting. To get you going quickly, it displays the following Sources and Destinations straight out of the box:



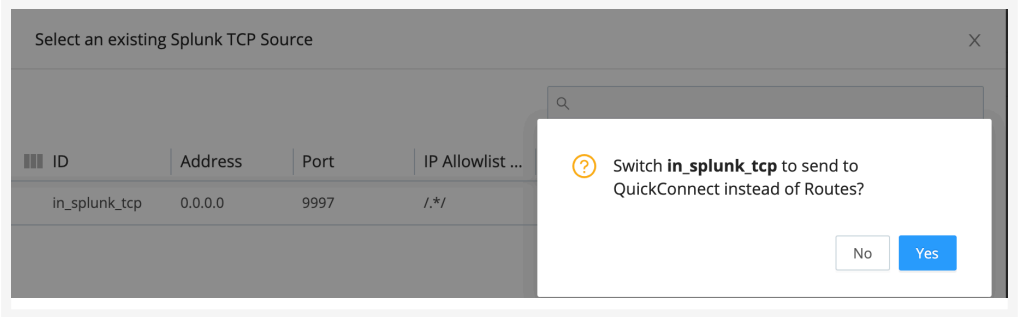
QuickConnect UI

Enabling Sources and Destinations

When you select Sources and Destinations, their configuration options will open in a drawer. Fill in a unique **Input ID**, and any other required fields marked by an asterisk (*). Then select **Save**.

Switching Contexts

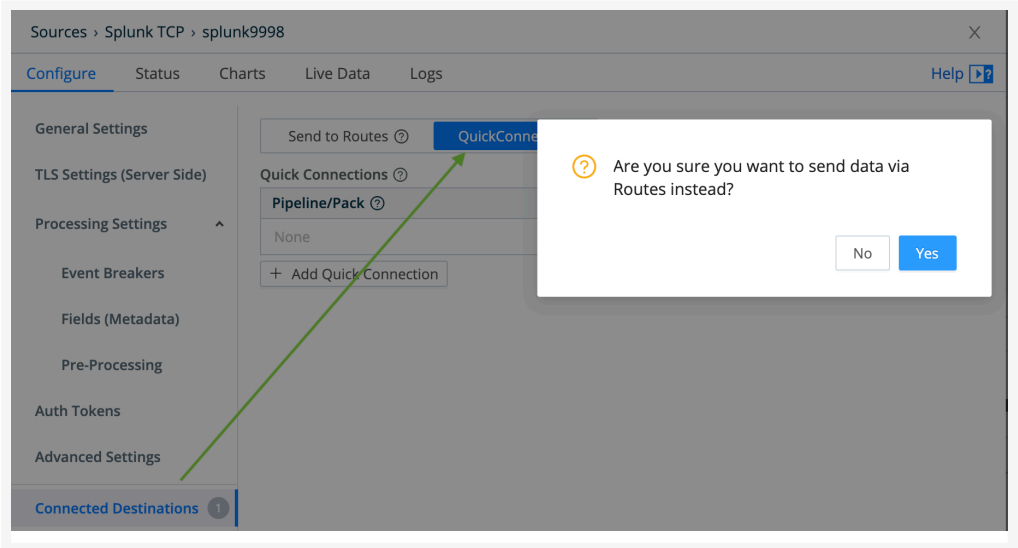
If you switch an **existing** Source to QuickConnect - for example, one of the preconfigured Sources that ship with Cribl Edge - you'll see the dialog shown below. This is due to QuickConnect's independence from the Routes interface - you need to confirm your choice to move the existing configuration from the Data Routes world to the QuickConnect world.



Switching a Source/Destination between worlds

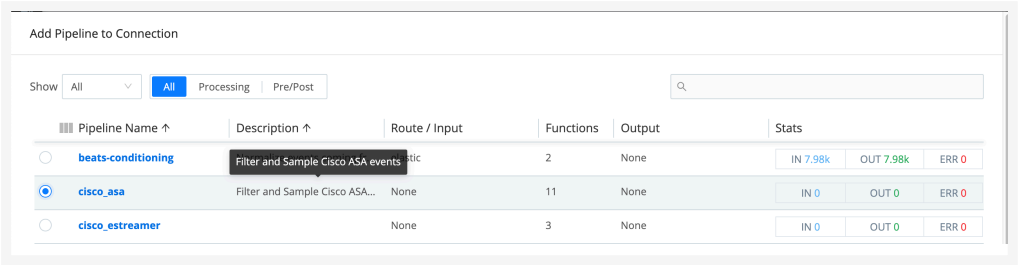
If you switch an existing Source config to QuickConnect, its data will no longer flow through its configured Route.

You can also go the opposite way: move a Source originally configured in QuickConnect to the Routing interface. In the Source's drawer or config modal, select the **Connected Destinations** left tab, and then select the resulting **Send to Routes** button. Here again, you'll need to confirm your choice before proceeding.



Selecting a Pipeline or Pack

When you add or modify a connection line, the bottom **Connection Configuration** drawer will prompt you to select a **Passthru**, **Pipeline**, or **Pack** connection. Selecting either **Pipeline** or **Pack** will open an **Add…** modal like this in the bottom drawer:



QuickConnect versus Routing UIs

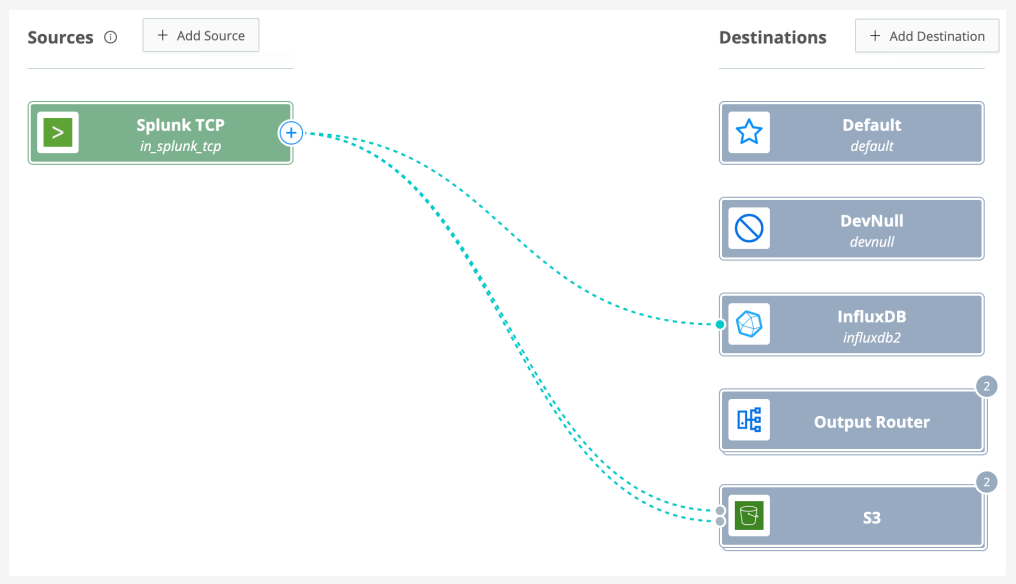
Here, you're selecting among already-configured Pipelines or Packs. **Resist the temptation to select a bold blue Pipeline or Pack name.** (Doing so will open the Pipeline or Pack config in a separate browser tab, which might offer more configuration than you bargained for.)

Instead, select the subtle radio button to the left of the Pipeline or Pack you want to open. Then select **Save** to confirm your choice and close the drawer.

Multiple Connections

You can drag multiple connection lines between a given Source/Destination pair. (You might do this, for example, to configure parallel Pipelines to handle different data types.)

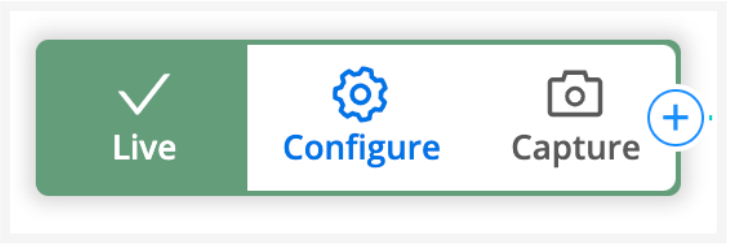
Also, you can connect one Source to multiple Destinations, or multiple Sources to one Destination. Let's isolate examples of both, from the screen capture we started with:



Multiple connections out, multiple connections in

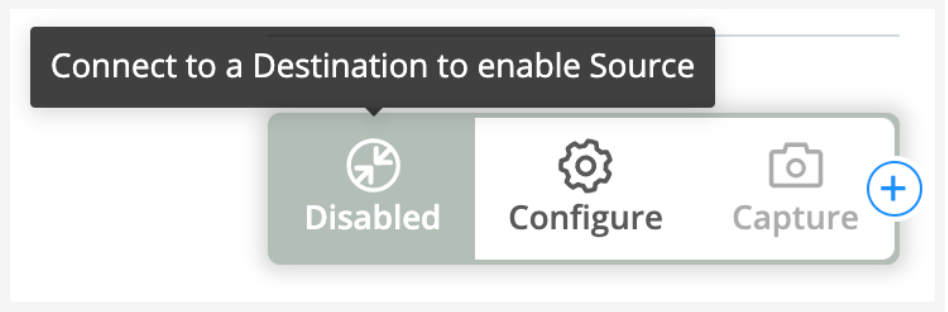
Source/Destination Tiles' Hover Options

Once you've configured a Source or Destination, hovering over its tile will display options like these:



Signed, sealed, delivered: tiles' on-hover options

The leftmost button is an indicator of the Source's/Destination's state. Live means healthy; disabled, error, or warning states will also appear here. Note that a Source will appear as **Disabled** until you connect it to a Destination.



This Source is not misconfigured - it's just lonely

To update the Source or Destination configuration or address any problems, select the middle **Configure** button.

The right **Capture** button captures a sample of data flowing through the Source or Destination. This is visualized in a drawer version of the Live tab.

Group Connections

Once you've configured two or more copies of the same Source or Destination type

- for example, to support different configurations of one integration or protocol - QuickConnect stacks their tiles together to keep the display clean.



1.4. Onboard Faster With Packs

As you get started with Cribl Edge, consider using Packs to help streamline your onboarding process and get more value out of your Cribl deployment quickly.

Packs are self-contained, portable bundles of Cribl objects with pre-built logic that solves typical or specific data processing challenges. They can include configurations or example logic for a variety of objects, such as Routes, Pipelines, and Knowledge objects.

On the Cribl Packs Dispensary, Cribl officially publishes and maintains a variety of validate Packs that you might use to solve a particular challenge or process data from a variety of vendors, including many that do not yet have full Cribl Edge integrations. The Dispensary also hosts validated Packs submitted from the larger Cribl developer and user community.

Search and find for Packs that match your specific use case. Then download and import these Packs to:

- **Reduce manual effort for common vendors and log formats:** Find Packs for validated solutions created by Cribl employees and community members to skip the time-consuming process of writing complex logic for well-known log formats. Instead, import a Pack to begin ingesting, parsing, and enriching logs for common vendors quickly.
- **Learn Cribl best practices:** Use configurations created by experts from Cribl and the community to learn how to build an effective solution for common data processing challenges.
- **Get value from Cribl more quickly:** Move from initial source connection to usable, parsed data more quickly by using pre-programmed logic in Packs that you can adapt for your specific needs.
- **Simplify complex collections:** Use REST Packs to streamline configurations for the REST Collector. These officially supported Packs eliminate manual effort by providing validated authentication, pagination, and event-breaking settings tailored to specific upstream services that do not yet have a dedicated Source or Collector.
- **Perform common data processing tasks:** Use Packs for use cases such as normalizing timestamps, flattening JSON objects, or masking sensitive PII.



This page describes how to use Packs to accelerate the onboarding process with Cribl Edge. However, you can also use Packs for advanced environment templating as your Cribl Edge deployment becomes more complex and needs to scale. See Pack-Based Configuration Management for more information about this advanced configuration management use case.

1.4.1. Get Pre-Built Pack Content

Packs created by Cribl and the Cribl Community let you speed up configuring your data flows by providing ready-made solutions for many common integrations and use cases.

On the Cribl Packs Dispensary, Cribl officially publishes and maintains a variety of validate Packs that you might use to solve a particular challenge or process data from a variety of vendors, including many that do not yet have full Cribl Edge integrations. The Dispensary also hosts validated Packs submitted from the larger Cribl developer and user community.

Packs created by Cribl and the Cribl Community let you speed up configuring your data flows by providing ready-made solutions for many common integrations and use cases. These Packs are hosted on the Cribl Pack Dispensary. You can download and customize each Pack as needed.

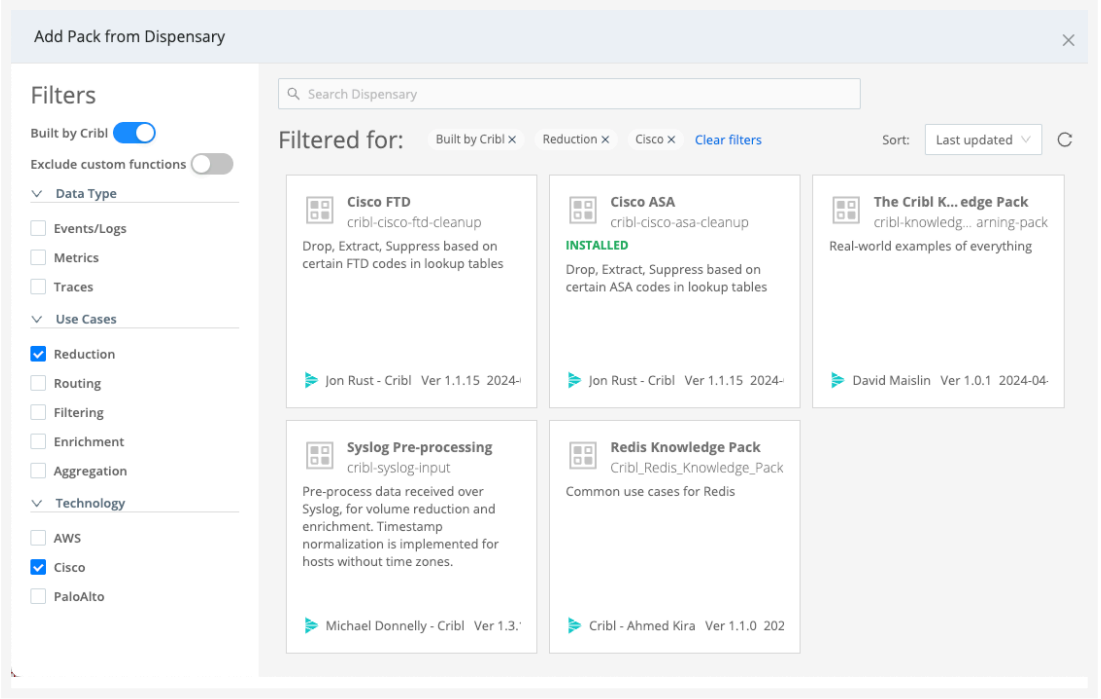


This page describes how to use Packs to accelerate the onboarding process with Cribl Edge. However, you can also use Packs for advanced environment templating as your Cribl Edge deployment becomes more complex and needs to scale. See Packs for more information about this advanced configuration management use case.

Add a Pack From the Dispensary

To import a Pack from the Dispensary directly into your deployment:

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet.
2. Navigate to **Processing**, then **Packs** and select **Add Pack**. From the submenu, select **Add from Dispensary**.
3. The Packs Dispensary will open in a drawer.

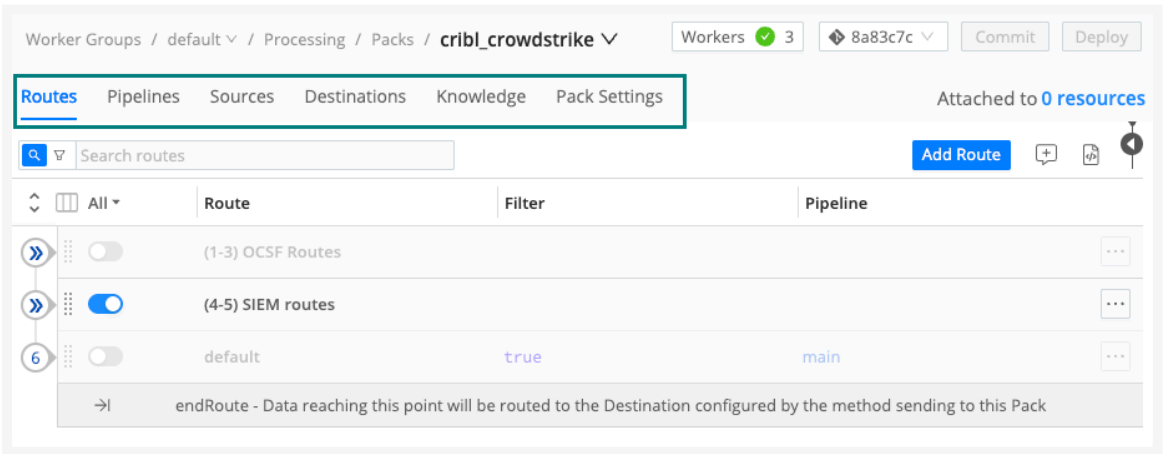


Cribl Packs Dispensary

4. Using the drawer controls, browse or search for the Pack(s) you want. (You can use the check boxes at the left to filter by data type, use case, and technology.)
5. Select any Pack tile to display its details page with its README. This will typically outline the Pack's purpose, compatibility, requirements, and installation.
6. To proceed, select **Add Pack** on the Pack details page.
7. If the Pack you are importing binds variables to any fields, you will see a **Configure Variables** button in the notification about successful import.
   Select it to go to the **Variables** page, where you can configure the values for the variables for your deployment.

Explore and Customize the Pack

Click inside the new Pack. In the Pack submenu you can see all the components that the Pack contains: Routes, Pipelines, Sources, Destinations, and Knowledge Objects.



Pack menu showing Routes, Pipelines, Sources, Destinations, and Knowledge Objects

At this point you can edit any elements in the Pack to customize it to your needs. Do it the same way you would when working directly in any Fleet.

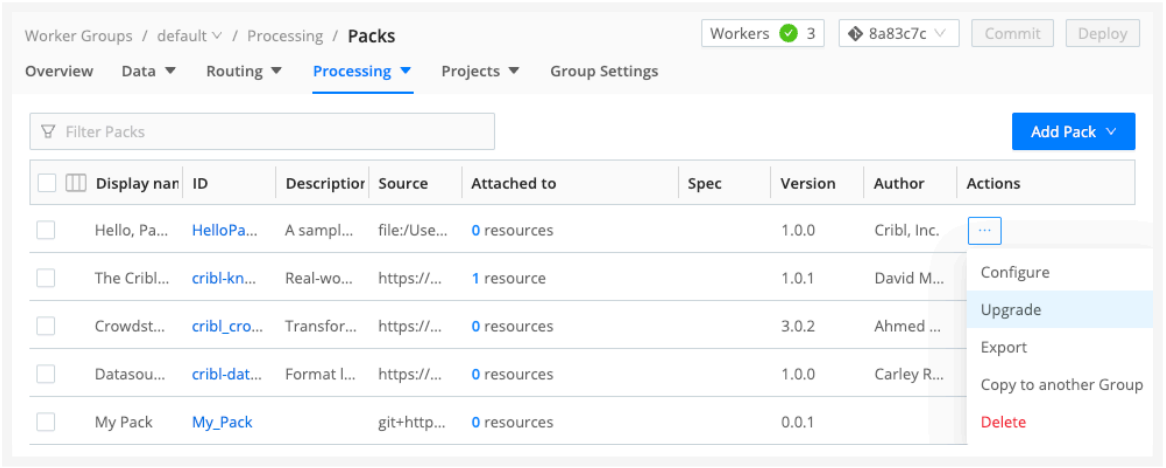


Remember that all elements inside the Pack, including Knowledge Objects, are scoped to that Pack. This means you cannot use, for example, a Knowledge Object defined inside a Pack outside it.

Upgrade a Pack from the Dispensary

When a new version of a Pack is published to the Dispensary, you can upgrade your local Pack to get all the latest improvements.

If you didn't introduce local changes to the Pack, you can upgrade the Pack directly from the UI. On the **Packs** page, open the Options ••• menu for your Pack, and then select **Upgrade**.



Upgrading an existing Pack

If you've modified the Pack, Cribl Edge will not overwrite it during upgrading, to prevent deletion of your locally created resources. To reconcile your own changes with improvements coming in from a new Pack version:

1. Import the updated Pack as a separate Pack with a new name.
   Give it a name that includes the version number (for example, `cribl-syslog-input-120`).
2. Do a side-by-side comparison of the previous and new versions of the Pack.
   Copy configurations from your modified Pack into the upgraded Pack.
   Review all comments in the upgraded Pack to understand changes introduced in the new version.
3. Enable or disable any Functions in the new Pack as necessary.
4. Update any Routes, Pipelines, Sources, or Destinations that use the previous Pack version to reference the new Pack.

5. If you modified default Cribl Edge Knowledge Objects (for example, lookup files), copy the modified files locally for safekeeping, before upgrading the Pack.
   After you install the upgrade, copy those files back to the upgraded Pack, overwriting the default versions in the Pack.
6. Commit and Deploy.

Troubleshoot Packs from the Dispensary

If you experience unexpected behaviors using Packs from the Dispensary, the following troubleshooting steps can help you resolve the issue quickly. If you continue to have trouble, reach out to Cribl Support.

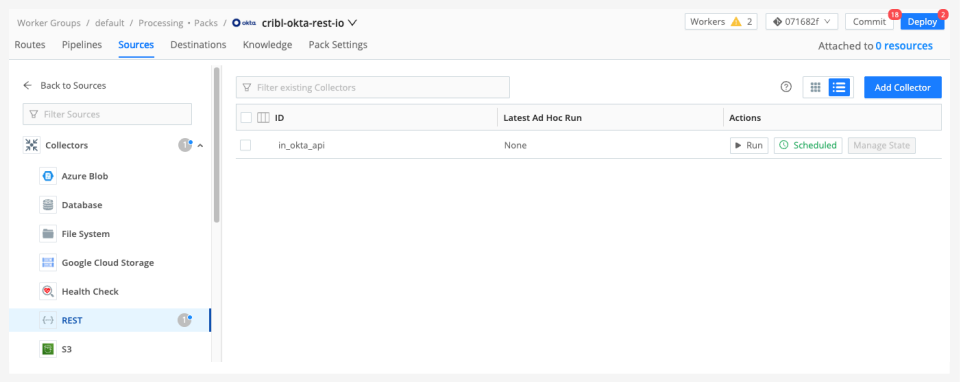
Imported Pack Does Not Appear in Routes

If a Pack downloaded from the Dispensary does not appear in the **Pipeline** dropdown of your Routing table at the Fleet level, that is likely because the Pack contains either a Source or Destination.

Only Packs scoped exclusively for data processing appear in the **Pipeline** dropdown. Packs that include a Source or Destination are treated as independent integrations, which are intended for end-to-end data flows or as specialized Source-side or Destination-side components. See Purpose and Scope of the Pack for a deeper explanation of how Packs can be scoped.

First, verify that the Pack has a Source, Collector, or Destination:

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet.
2. Navigate to **Processing**, then **Packs**. Select the name of the Pack from the list.
3. From inside the Pack, select either **Sources** or **Destinations** from the top bar to see the list of configured Sources or Destinations. Verify that the Pack has a Source or Destination, which means it will have a badge with a number that indicates how many Sources, Collectors, or Destinations are contained in that Pack.



Example of a REST Collector contained in a Pack

If the Pack does not contain a Source, Collector, or Destination, there is likely some other issue. However, if it does contain one, the next step is to route data to or from that Pack, depending on whether it contains a

Source, a Destination, or both.

2. Explore Cribl Edge

Explore the Cribl Edge UI.

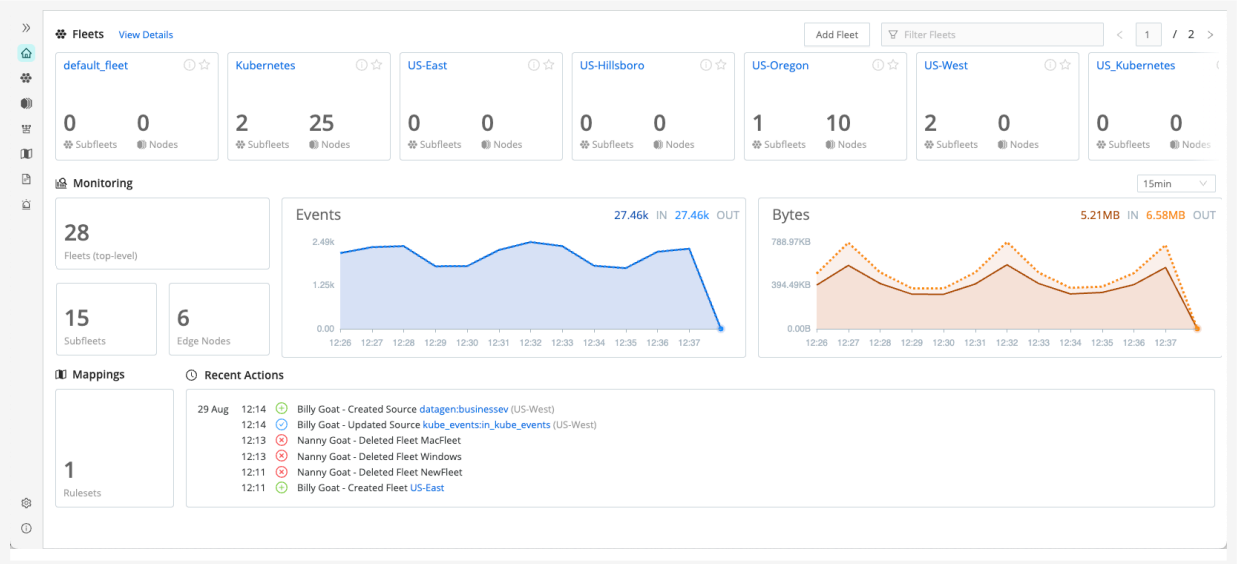
The Cribl Edge UI offers a centralized view to manage, configure, and version-control your Edge Nodes. It also lets you teleport to individual Nodes to locally preview and validate your configurations. Here's a tour of the Cribl Edge UI in Distributed mode.



The available UI options might differ depending on your deployment type (Cribl.Cloud versus on-prem) and the platform the Edge Node is installed on (Linux versus macOS versus Windows).

Edge Home

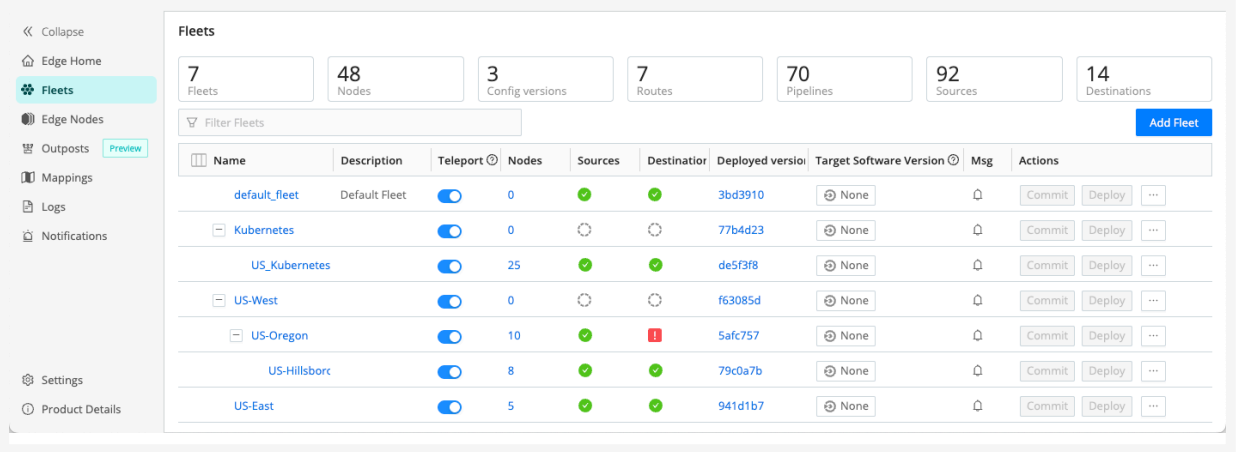
When you first log into Cribl Edge you'll land on the **Edge Home** page. This page gives you an overview of configured Fleets, monitoring information for Fleets, Subfleets, Edge Nodes, and Events and Bytes in and out. At the bottom, you'll see Recent Actions that users have taken in the environment as well as any configured Fleet Mappings.



The Edge Home page

Fleets

Select **Fleets** from the sidebar to access your configured Fleets.



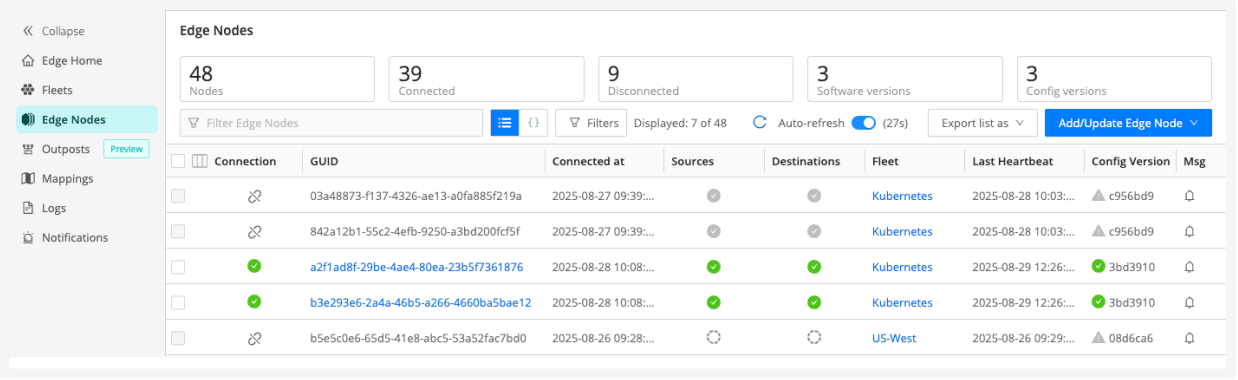
Fleets page

From here, you can select a Fleet name to isolate individual Fleets, or use the filter bar to locate your Fleet.

- For more information about the Fleets interface, see Fleets.
- To learn about creating Fleets, see Creating and Managing Fleets and Subfleets.

Edge Nodes

Select **Edge Nodes** to view all of your Edge Nodes in one list.



Edge Nodes page

The list provides status information for each Edge Node in the selected Fleet, lets you filter Edge Nodes, add, and update Nodes. For more information, see Manage Edge Nodes.

Outposts

Select **Outposts** to view the list of configured Outposts, add and manage them. For more information, see Connect Through Cribl Outpost.

Mappings

You can use Mappings Rulesets to map Edge Nodes to Fleets by defining rules. For more information, see Mapping Edge Nodes to Fleets.

Logs

Cribl Edge generates internal application logs that monitor its own operations and health. They provide valuable insights into the system's behavior, performance, and potential issues. For more information, see Internal Logs.

Notifications

Notifications alert Cribl Edge admins about issues that require their immediate attention. For information on types of Notifications, Notification targets, and managing Notifications, see Managing Notifications.

Explore Tab

The **Explore** tab lets you get details about a particular Edge Node.

For more details about the information you can find in the **Explore** tab, see Explore Cribl Edge Nodes.

2.1. Explore Cribl Edge Nodes

Examine Edge Nodes details in the Explore tab.

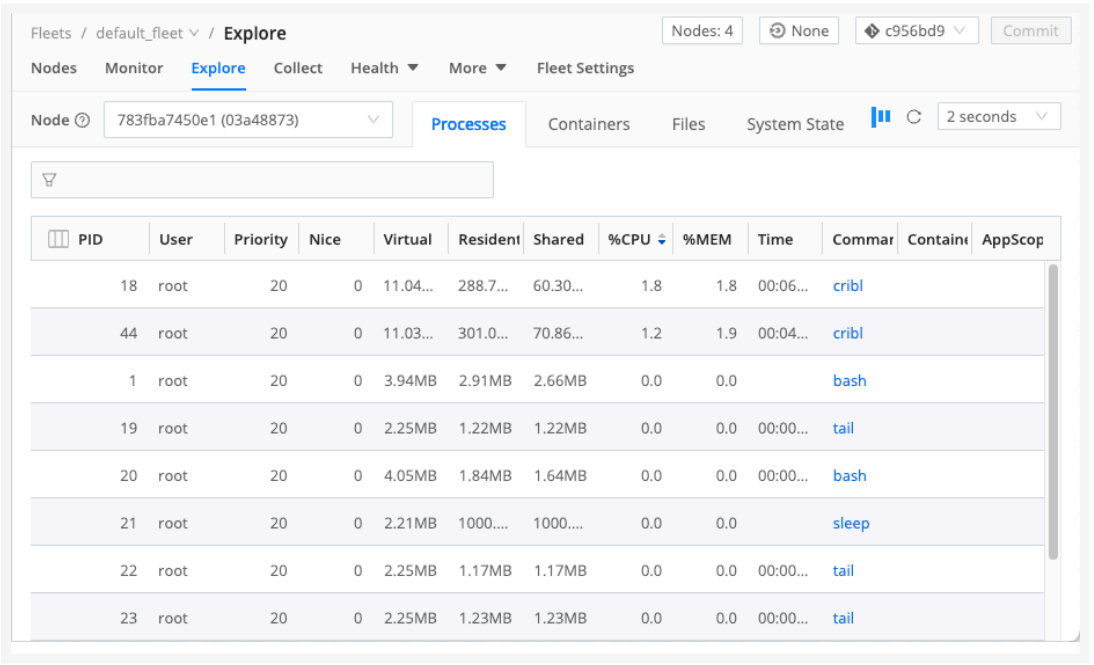
You can view details of each individual Edge Node by using the **Explore** tab.

To access it, in the sidebar, select **Fleets**, then choose a Fleet and select **Explore**.

In the **Node** drop-down, select one of your hosts to display the following tabs:

- Processes
- Containers
- Files
- System State
- Kubernetes

The available tabs will depend on your deployment environment.



Explore an Edge Node

The **Node** drop-down lists up to 100 Edge Nodes, ordered by hostname. To view the details for a specific Edge Node, enter the hostname or GUID into the **Node** field.

Let's explore each tab in the **Explore** view.

Processes

The **Processes** tab lists all the processes running on the Edge Node.

Select any of the rows to display a drawer with basic information on the process. In the drawer's default **Overview** tab, you'll find information such as CPU, Memory, and IO graphs, along with tables for active **Listening**, **Inbound**, and **Outbound** connections, if applicable.

Select **All details** to see the selected process' information out of `/proc`, expressed as key-value pairs. To access this information, you would normally need to SSH to the machine. This view makes troubleshooting across multiple systems much easier.

Process Metrics on Windows

In Windows-based Fleets, you can control the collection method for Process Metrics using a configuration option in the **Fleet Settings** > **Limits** > **Other** section. This setting allows you to choose between using PowerShell cmdlets or the more efficient native module.



When using PowerShell cmdlets on Windows, the environment variables displayed in this tab might be inaccurate. For details, see Other Limits.

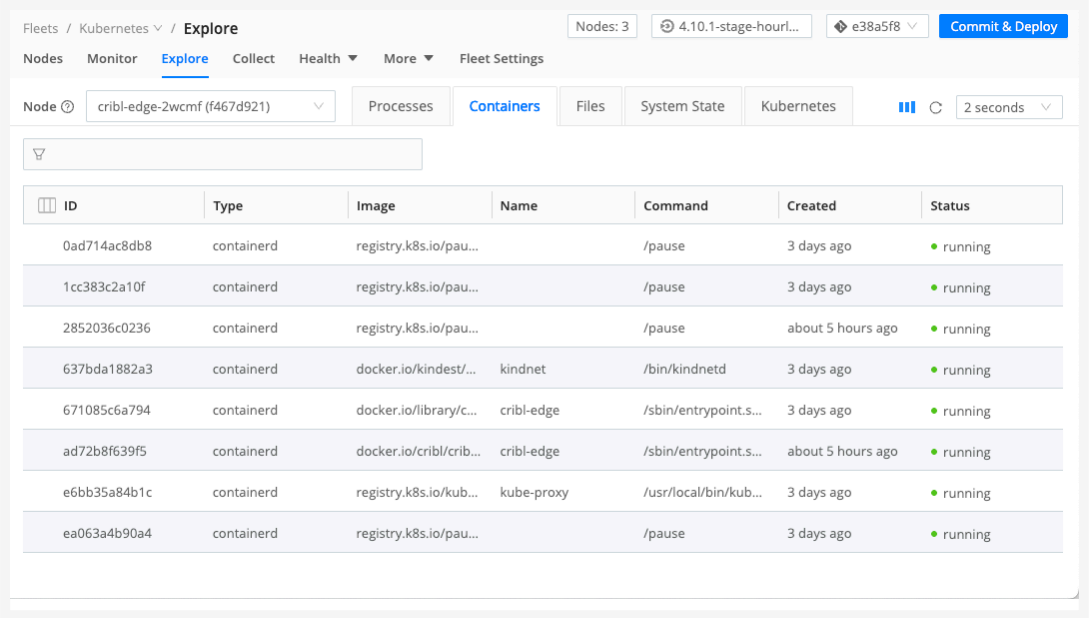
Containers

The **Containers** tab, available for Edge Nodes running in containers, lists all the running containers and container metrics including information about images, volumes, status, ports, and so on.



Cribl Edge supports both Docker and `containerd` runtimes.

`containerd` containers have less info than Docker containers, so `Ports`, `IPs`, and `Logs` won't populate.



Containers overview

Select any container to open a drawer with more details, including container logs.

If you run Edge as an unprivileged user, see Making Docker Containers Visible to Edge.

Files

The **Files** tab lists all the log files being actively written to by running applications that Cribl Edge has auto-discovered. You can also specify a list of directories and files to actively monitor.

The **Actions** column allows you to:

- **View**: Displays a representation of the lines this column contains. You can also select any file row. To restrict how much data is displayed, use the search field or time picker on the **Search** tab.
- **Inspect**: Opens the **Inspect File** tab to show file metadata, including details like permissions, file size, user, and modified date.
  When inspected, compressed files (such as `foo.bar.gz`) include a **File preview** that shows the beginning of the file contents.
  Archived files (such as `.zip`, `.tgz`, and `.tar.gz`) include a **File listing** that shows the files within it.
  If a file appears suspicious, select **VirusTotal** or **OpSwat** at the bottom of the modal to see whether the file is flagged as compromised.
- **Monitor**: Displays the File Monitor's configuration modal.
- **Ingest**: Opens the Ingest file modal to send the file content to Routes/Pipelines for further processing or downstream to any destination you have configured. This is useful for testing and troubleshooting your configurations.

The **Files** tab provides the following options.

File Discovery Modes

Select a button at the top to select a discovery mode:

- **Auto**: Tells Cribl Edge to automatically discover files that are open for writing on currently running processes. The **Auto** discovery mode is unavailable on Windows and macOS.
- **Manual**: Tells Cribl Edge to discover the files within the **Path** (directory) and **Allowlist** that you specify, down to the **Max depth**.
- **Browse**: Displays a tree view of all of your directories and files.

Path

The **Path** field tells Cribl Edge to discover the files within the path (a directory) that you specify, down to the **Max depth**.

Allowlist

The **Allowlist** field, available with Auto and Manual discovery, supports wildcard syntax, and supports the exclamation mark (`!`) for negation. For example, you can use `!*cribl*access.log` to prevent Cribl Edge from discovering its own access log. The default filters are `*/log/*` and `*log`.

Select any file to see a representation of the lines it contains. To restrict how much data is displayed, you can use the search field or time picker on the **Search** tab.

If the representation of events shown on the **Search** tab isn't ideally suited to the file's contents, you can use the **Event Breakers** tab to refine it.

Monitor Files

The **Monitor Files** button, available with Auto and Manual discovery, opens a new File Monitor Source. The Source is prefilled with the discovery mode and anything else you specified on the **Files** tab, such as allowlist entries, path, or max depth.

Max Depth

The **Max depth** field, available with Manual discovery, is empty by default. Cribl Edge will search subdirectories, and their subdirectories, downward without limit.

If you enter `0`, Cribl Edge will discover only the top-level files within the specified path. If you specify `1`, Cribl Edge will discover files one level down from the top. Follow this pattern to specify the depth you want.

Monitor a File

Select a file's **Monitor** or **Actions** option to configure your File Monitor Source to generate events from the file's lines or records.

The **Monitor** feature automatically populates the modal with the following settings configured on the **Files** tab:

- **Discovery mode**. Defaults to **Manual** for Windows and macOS.
- **Search path**
- **Max depth**
- **Filename allowlist**

In addition, the **Connected Destinations** section defaults to **QuickConnect**. In the **Connected Destinations** section, you can select a Pipeline or Pack and a Destination. Otherwise, when you save, you'll be routed to the **Collect** page to set up your connections via QuickConnect. For details about making these connections, see File Monitor and QuickConnect.

Ingest a File

To configure options for how and where to send file contents, use the **Ingest file** modal.

You have two options:

- Send directly to a configured Destination via **QuickConnect** (the default).
- Send to Routes through (an optional) Pre-Processing Pipeline.

You can configure Event Breakers and rulesets for both options.

Explore Files with Event Breakers

When you select a file in the **Files** tab, and Cribl Edge shows a representation of the lines that the file contains, how does that work? What's happening is that Cribl Edge is applying a default Event Breaker to the file.

You are not limited to the default Event Breaker, though. Select the **Event Breakers** tab, then:

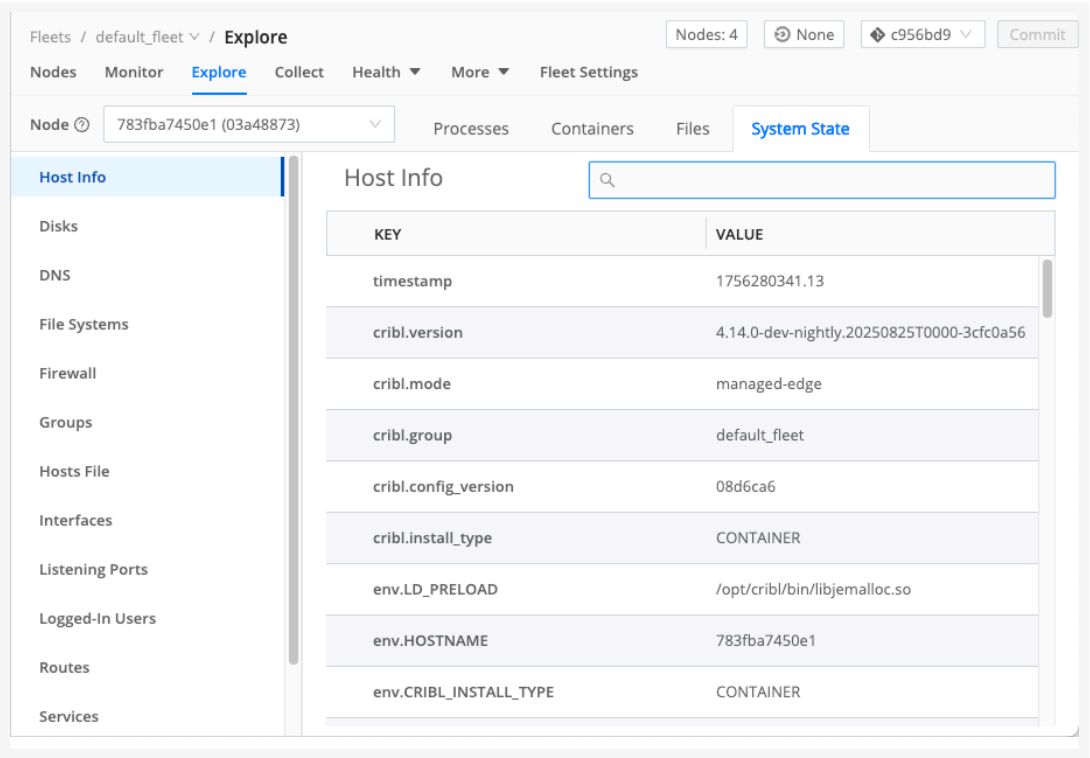
- To apply a different (existing) Event Breaker, select **Add ruleset**, then select the desired ruleset from the **Event Breaker rulesets** drop-down.
- To create a new Event Breaker Ruleset, click **Create New**. In the resulting **New Ruleset** modal, proceed as described here. Later, you can reuse the new Event Breaker as part of a Source or a Collector.
  While you create the new ruleset, Cribl Edge pulls the contents of the open file into the Sample File area. Toggle between the **In** and **Out** tabs to compare, respectively, the original content and the content as modified by the Event Breaker you're creating.

Now return to the **Search** tab - the contents of your chosen file will appear with the new Event Breaker applied.

System State

The **System State** upper tab provides access to a collection of information gathered by the System State Source.

To display any of the information, you need to configure and enable the System State Source. Also, make sure that the Source's Collector Settings fields are enabled.



System State for the selected Edge Node

Host Info

Cribl Edge can add a `__metadata` property to every event emitted from every enabled Source. The **System State** tab displays the metadata collected for each Edge Node under **Host Info**.

You can use the metadata surfaced by an Edge Node to:

- Enrich events (with an internal `__metadata` field).
- Display to users as a part of instance exploration.

To customize the type of metadata collected, select **Fleet Settings** > **Limits** > **Metadata**. Use the **Event metadata sources** drop-down (and/or the **Add source** button) to add and select metadata sources.

In Edge mode, all the **Event metadata sources** are enabled by default.

The metadata sources that you can select here include:

- `os`: Reports details for the host OS and host machine, like OS version, kernel version, CPU, and memory resources, hostname, network addresses, and so on.
- `cribl`: Reports the Cribl Edge version, mode, Fleet for managed instances, tags defined on the instance, and config version.
- `aws`: Reports details for an EC2 instance, including the instance type, hostname, network addresses, tags, and IAM roles. For security reasons, we report only IAM role names.
- `env`: Reports environment variables.
- `kube`: Reports details on a Kubernetes environment, including the node, Pod, and container. For details, see __metadata.kube Property.

When these metadata sources are enabled (and can get data), Cribl Edge will add the corresponding property to events, with a nested property for each enabled source.

Some metadata sources work only in configured environments. For example, the `aws` source is available only when running on an AWS EC2 instance.

If your security tools report denied outbound traffic to IP addresses like `169.254.169.168` or `169.254.169.254`, you can suppress these by removing `aws` from the metadata sources described above. If you have a proxy setup, Cribl recommends adding these IP addresses to your `no_proxy` environment variable.

`__metadata.kube` Property

For the `__metadata.kube` property (`kube` in the UI) to report details on a Kubernetes environment, Cribl Edge needs to figure out where it is running. Set the `KUBE_K8S_POD` environment variable to the name of the Pod in which Cribl Edge is running. At this point, the `__metadata.kube` property will have information to report on the `node` and `pod` properties.

If the `/proc/self/cgroup` is working, then the `container` property information will be available, too.

If you leave the `KUBE_K8S_POD` environment variable unset, and `/proc/self/cgroup` is not working, then Cribl Edge will not know what Pod it is running in. This state has multiple implications:

- The Kubernetes Metrics Source will be unable to identify whether or not it is in a DaemonSet. The result will be redundant metrics from each node in the cluster.

- The Kubernetes Metadata Collector will not add the `__metadata.kube property`.
- The Kubernetes Logs Source will also duplicate data. Every node in the cluster will collect logs for every container in the cluster.

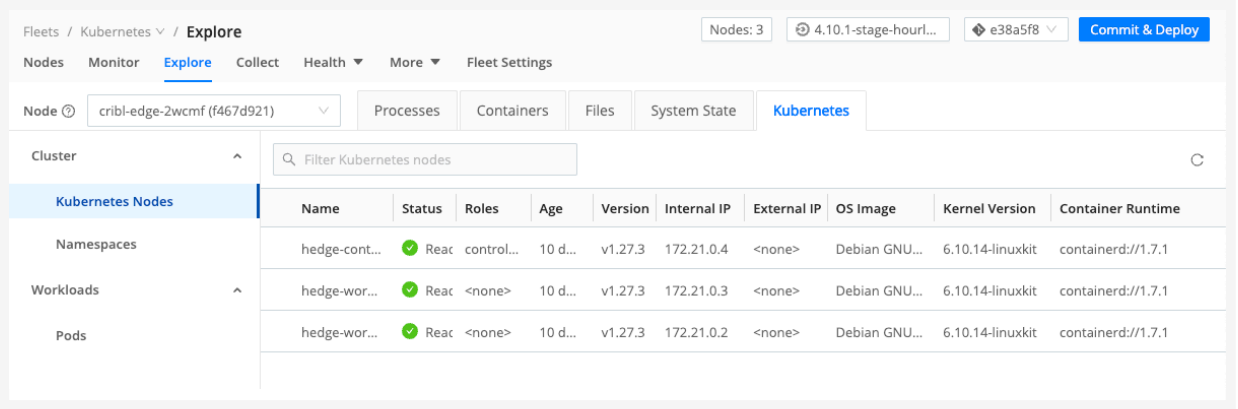
Other System State Information

The remaining information in the System State tab is organized into the following tabs:

Tab	Description
Disks	The **Disks** tab displays the inventory of physical disks and their partitions on the host system.
DNS	The **DNS** tab lists the host system's DNS resolvers and search entries.
File Systems	The **File Systems** tab displays an inventory of the mounted file systems on the host system.
Firewall	The **Firewall** tab displays a list of the host's defined firewall rules.
Groups	The **Groups** tab displays a list of the local groups including their names, descriptions, and members on the host system.
Hosts File	The **Hosts File** tab displays the host system's current state.
Interfaces	The **Interfaces** tab displays a list of each of the network interfaces on the host system.
Listening Ports	The **Listening Ports** tab displays a list of listening ports and their associated process identifier (pid).
Logged-in Users	The **Logged-In Users** tab displays a list of currently logged-in users on the host.
Routes	The **Routes** tab displays entries from the network routes on the host system.
Services	The **Services** tab displays a list of each configured service (such as `systemd` and `initd`) along with their running status.
Users	The **Users** tab displays a list of local users on the host system.

Kubernetes

The **Kubernetes** tab, available only for Kubernetes-based Edge Nodes, lists all the Kubernetes clusters, including nodes, pods, and containers.



Kubernetes tab

Select any row to open a summary table for the pod, including:

Tab	Description
Overview	Basic node information, including creation time, labels, taints, lease, and annotations.
Host	Detailed information about the node, including its IP address, capacity, system information, and the number of pods it's currently hosting.
Pods	List of pods running on the node, along with their resource usage and age.

Select **Filter Rules** to view/add more filters. The default filter ignores the `kube-*` namespace, which is typically used by Kubernetes system components and may contain a large volume of irrelevant events.

Filter Rules can be applied to the Kubernetes Logs Source to refine the data ingested into Cribl Edge.

To preview the effects of a new or modified Event Breaker Ruleset, you must commit and deploy the configuration changes.

Kubernetes Namespaces

In the sidebar, select **Namespaces** to view the list of active namespaces in the Kubernetes cluster, along with their status and age.

Select any namespace to view detailed information, including creation time, labels, annotations, resource quotas, and resource limits.

Kubernetes Pods

In the sidebar under **Workloads**, select **Pods** to view a list of pods running in the cluster. For each pod, the list shows its associated name, namespace, readiness, status, restart count, age, IP address, and the node it's

running on.

Select any row to view pod information with the following tabs available:

Tab	Description
Overview	General information about the pod, including its name, namespace, status, IP address, labels, annotations, current health conditions, and tolerations.
	Details about the containers within the selected pod, including their name, image, ports used

3. Cribl Edge On Linux

Learn about deploying and configuring Cribl Edge on Linux.

Cribl Edge on Linux offers easy-to-use tools for exploring and collecting events. You can run Cribl Edge on a supported Linux and collect events, logs, and metrics via the full offering of Cribl Edge Sources.

Read the following topics for more information about running Cribl Edge on Linux:

Topic	Description
Installing Cribl Edge on Linux	Basic Cribl Edge deployment requirements and procedures
Enabling Start on Boot	Enable Cribl Edge instances to restart on system boot
Running Edge as an Unprivileged User	Install Cribl Edge an unprivileged user
Installing Edge on Linux via RPM	Securely install Edge using an RPM package
Installing Edge on Linux via .deb	Install Cribl Edge from a Linux .deb package
Anti-Virus Exceptions	Add anti-virus exceptions to facilitate Cribl Edge installation
FIPS Mode for Cribl Edge (Linux)	Running Cribl in FIPS mode
Use ACLs to Allow Cribl Edge to Read Files	Using Access Control Lists to allow Cribl Edge to Read Files

3.1. Install Cribl Edge On Linux

Install and deploy Cribl Edge Nodes on-prem on Linux.

You can install Cribl Edge in a supported Linux environment either connected to a Leader Node, alongside Cribl Stream, or in multiple instances on one host.

You can also install a Linux Edge Node in a Distributed deployment by running the bootstrap script in your command prompt.

Minimum Requirements

The following requirements are the minimum for installing and running an Edge Node on Linux.



Below is a summary of requirements for installing Cribl Edge. See Requirements for more detailed information.

OS Requirements

Cribl Edge Nodes can be installed on any common Linux distribution that fulfils the following criteria:

- 64-bit kernel >= 3.10
- glibc >= 2.17
- OpenSSL >= 3.0 (if FIPS mode is enabled)

Application Requirements

Minimum application requirements for Edge Nodes are:

- ~1 Ghz processor
- 1024 MB RAM
- 5 GB free disk space per volume for operational purposes

Download Cribl Edge

1. First, download the install package, which is the same binary as Cribl Stream, to your Linux machine.

2. Next, ensure that required ports are available (see Ports).
3. Then, you can install and run any of the following:
    - Connected to a Leader Node.
    - Cribl Edge and Cribl Stream on the same host.
    - Multiple installations of the same product, on the same host.

Install Cribl Edge

These instructions explain how to run Cribl Edge locally on your own machine and connect it to a Leader Node.

1. Un-tar the install package in a directory of choice, and rename the resulting `cribl` directory to `cribl-edge`, for example:

```
cd /opt/
tar xvzf cribl-<version>-<build>-<arch>.tgz
mv /opt/cribl/ /opt/cribl-edge
```

Copy



To prevent issues with Cribl file operations, `/opt/cribl` and all its subdirectories must reside on the same device. Mounting separate devices within `/opt/cribl` is not recommended. For external storage needs, such as lookups or Persistent Queues (PQs), create a completely separate directory outside of `/opt/cribl`. While `/opt/cribl` is the default installation path, Cribl can be installed to other locations if necessary.

2. Set the renamed directory, (for example, `/opt/cribl-edge/`), as your `$CRIBL_HOME` directory.



To make the `$CRIBL_HOME` environment variable available on your command line, you can:

- Assign it once, using the `export` command:
  `export CRIBL_HOME=/opt/cribl-edge`
- Set it as a default, by adding it to your to your terminal profile file.

3. Change the ownership of the installation to run Cribl Edge as a non-privileged user:

```
chown -R cribl:cribl /opt/cribl-edge
```

Copy

4. Start the Cribl service:

```
/opt/cribl-edge/bin/cribl start
```

Copy

5. Set the installation mode to `mode-managed-edge` to connect the Node to the designated Leader:

```
/opt/cribl-edge/bin/cribl mode-managed-edge /
   -H <leader-hostname-or-IP> /
   -p <port> /
   -u <token> /
  [-g <fleet>]
```

Copy

`<token>` must match the Leader token, to ensure a connection with the Leader. Get the Leader token from **Settings** > **Global** > **System** > **Distributed Settings** > **Leader Settings** > **Auth token**.
The new Node will be assigned to a Fleet on the Leader, either `default_fleet`, or a different one according to Fleet mappings.
You can manually point to a specific Fleet by providing its name as `-g <fleet>`, but Fleet mappings will override this choice.



If you only need a Single-instance deployment of Cribl Edge (without a Leader), use `mode-edge` instead of `mode-managed-edge`. In that case you don't need to provide the Leader token.

6. Configure the installation as a service and restart it:

```
/opt/cribl-edge/bin/cribl boot-start enable -m systemd -u cribl
sudo systemctl restart cribl-edge.service
```

Copy



To find out more about the `boot-start` command, see Enabling Start on Boot.

7. Finally, go to `http://localhost:9420` (or another port you defined) and log in with default credentials (`admin:admin`). You can now start configuring Cribl Edge with Sources and Destinations, or start creating Routes and Pipelines.

For more ways to configure Leader and Edge Nodes, see Set Up Leader and Edge Nodes.

Cribl Edge and Cribl Stream on the Same Host

You can run an Edge Node with a Leader that is managing Cribl Stream, or an Edge Node and a Stream Worker Node on the same host. To accommodate these scenarios, each product has a distinct service name:

- `cribl-edge.service` for Cribl Edge.
- `cribl.service` for Cribl Stream.

Here, Cribl recommends un-tarring the download package twice, into two separate directories. This setup frees you to update and run each product individually. You could choose, e.g., `/opt/cribl-edge` and `/opt/cribl`:

```
cd /opt
tar zxvf /tmp/cribl-<version>-<build>-<arch>.tgz
mv cribl cribl-edge
tar zxvf /tmp/cribl-<version>-<build>-<arch>.tgz
```

Copy

The installation and configuration sequence will be the same for each product:

1. Change the ownership for both installations.
   Run Cribl Stream as a non-privileged user:

   ```
   chown -R cribl:cribl /opt/cribl
   ```

   Copy

   Run Cribl Edge as a non-privileged user:

   ```
   chown -R cribl:cribl /opt/cribl-edge
   ```

   Copy

Do NOT Run Cribl Edge as Root!

To listen on low ports 1-1024, Cribl Edge needs privileged access. You can enable this on systemd by adding this configuration key to your `override.conf` file:

```
[Service]
AmbientCapabilities=CAP_NET_BIND_SERVICE
```

Copy

If you want to add extra capabilities, such as reading certain resources (e.g., `/var/log/*`), add `CAP_DAC_READ_SEARCH` in a space-separated format as follows:

```
[Service]
AmbientCapabilities=CAP_NET_BIND_SERVICE CAP_DAC_READ_SEARCH
```

Copy

Alternatively, you can use ACLs to allow Cribl Edge to read files.

For details, see Running Edge as an Unprivileged User.

2. Set the correct mode, and configure each installation as a service. The `-H` and `-p` parameters are required.
   For Cribl Edge:

```
/opt/cribl-edge/bin/cribl mode-managed-edge -H <leader-hostname-or-IP> -p <port>

/opt/cribl-edge/bin/cribl boot-start enable

sudo systemctl restart cribl-edge.service
```

Copy

   For Cribl Stream:

```
/opt/cribl/bin/cribl mode-worker -H <leader-hostname-or-IP> -p <port> [options]

/opt/cribl/bin/cribl boot-start enable

sudo systemctl restart cribl.service
```

Copy



Managing IP Addresses

By default, Cribl Edge's API listens on port `9420`, instead of on Cribl Stream's default `9000` port. Some things to note:

- You can set the `CRIBL_EDGE` environment variable to any value to bind to `0.0.0.0`, instead of to the `127.0.0.1` address.
- If you are starting Cribl Edge from the CLI, make sure you set the `-H` parameter to `0.0.0.0`.
- If you connect your Edge Node to a Leader, the Node will automatically update the binding IP address to the one configured in the Leader's Fleet Settings. For an Edge Node to always listen on all IP addresses, you must update the Leader's **Host** value. (In the Leader's UI, go to **Fleet Settings** > **System** > **General Settings**, and change the **Host** field to `0.0.0.0`.)
- You can disable listening on the port by toggling **Listen on port** to off in **Fleet Settings** > **General Settings** > **Advanced**.

Continue with Setting Up Leader and Edge Nodes for Edge, and Setting Up Leader and Worker Nodes for Stream.

Multiple Installations of Same Product on Same Host

There are situations where it makes sense to run multiple Cribl Edge or Cribl Stream installations on the same host. For example, suppose two departments want to collect the data at the edge - and each wants to process the data differently, and to deploy its own Helm chart as a daemonset.

To support this: After un-tarring the installation package, copy or move it into a separate directory for each installation. For example, if you're creating two Cribl Edge installations:

```
cd /opt
tar zxvf /tmp/cribl-<version>-<build>-<arch>.tgz
cp -r cribl/ cribl-edge-01/
mv cribl/ cribl-edge-02/
```

Copy

The installation and configuration procedure is the same as described for collocated Cribl Edge and Cribl Stream above, except that you must:

- Omit steps pertaining to the product you are **not** installing.
- Repeat steps pertaining to the product you are **are** installing - once for each instance of the product.

Ensure that each instance of the product runs on its own dedicated port. Either:

- Specify different ports when you run the `mode-managed-edge` or `mode-worker` command; or,

- On the host, set the `CRIBL_AUTO_PORTS` environment variable to `1`.

Troubleshooting Resources

The Cribl University Troubleshooting course on Switching from Cribl Edge to Cribl Stream demonstrates these techniques for integrating Edge with Stream.

3.2. Enable Start On Boot On Linux

Ensure that Cribl Edge starts at boot time.

Cribl Edge ships with a CLI utility that can update your system's configuration to start Edge at system boot time. The basic format to invoke this utility is:

```
sudo $CRIBL_HOME/bin/cribl boot-start [enable|disable] [options] [args]
```

Copy

You must run this command as root, or with `sudo`. For options and arguments, see the CLI Reference.

The script will create a user named `cribl` to install, own, and run Cribl Edge/Stream.

Most, if not all, popular Linux distributions use `systemd` now to start processes at boot, while older or more obscure distributions may still use `initd`. Verify with your Linux distribution vendor if you aren't sure which method your systems use in order to know which procedure listed below to follow.

Using systemd

To **enable** Cribl Edge to start at boot time with **systemd**, you must run the `boot-start` command. Make sure you first create any user you want to specify to run Edge. E.g., to run Edge on boot as existing user `cribl`, you'd use:

```
sudo $CRIBL_HOME/bin/cribl boot-start enable -m systemd -u cribl
```

Copy

This will install a unit file (as shown below) named `cribl-edge.service`, and will start Cribl Edge at boot time as user `cribl`. A `-configDir` option can be used to specify where to install the unit file. If not specified, this location defaults to `/etc/systemd/system/`.

If necessary, change ownership for the Cribl Edge installation:

```
sudo chown -R cribl $CRIBL_HOME
```

Copy

Next, use the `enable` command to ensure that the service starts on system boot:

```
sudo systemctl enable cribl-edge
```

Copy

To **disable** starting at boot time, run the following command:

```
sudo $CRIBL_HOME/bin/cribl boot-start disable
```

Copy

Other available `systemctl` commands are:

```
sudo systemctl [start|stop|restart|status] cribl-edge
```

Copy

Note the file's default `65536` hard limit on maximum open file descriptors (known as a `ulimit`). The minimum recommended value is `65536`. Linux tracks this per user account. You can view the current soft `ulimit` for max open file descriptors with `$ ulimit -n` while logged in as the same user running the `cribl` binary.

Installed systemd File

```
[Unit]
Description=Systemd service file for Cribl Edge.
After=network.target

[Service]
Type=forking
User=cribl
Restart=always
RestartSec=5
LimitNOFILE=65536
PIDFile=/install/path/to/cribl/pid/cribl.pid
ExecStart=/install/path/to/cribl/bin/cribl start
ExecStop=/install/path/to/cribl/bin/cribl stop
ExecReload=/install/path/to/cribl/bin/cribl reload
TimeoutSec=60

[Install]
WantedBy=multi-user.target
```

Copy

Persisting Overrides

By default, disabling and re-enabling boot start will regenerate the `cribl-edge.service` file. To persist any overrides - such as proxy or privileged port usage - use this command:

```
sudo systemctl edit cribl-edge
```

Copy

This opens a text editor that prompts you to enter overrides, then saves them to a persistent file at:

```
/etc/systemd/system/cribl-edge.service.d/override.conf
```

Copy

Using a Literal % Character

To use a literal `%` character, make sure to escape it by using a double `%%`.

For example, suppose that you want to persist the environment variable `CRIBL_DIST_WORKER_PROXY` in a systemd unit file, and the value includes a password that contains the special character `@`: `special@password`. You must encode the `@` as `%40`. However, because `%` is a special character that is used for specifiers in systemd unit files, you must also escape the `%` itself by doubling it.

In this example, `%%40` represents the literal `@` character in the password:

```
Environment=CRIBL_DIST_WORKER_PROXY=socks5://proxyuser:special%%40password@mydomain.c
```

Copy

Avoid Running Cribl Edge as Root

Do not run Cribl Edge as root!

To listen on low ports 1-1024, Cribl Edge needs privileged access. You can enable this on systemd by adding this configuration key to your `override.conf` file:

```
[Service]
AmbientCapabilities=CAP_NET_BIND_SERVICE
```

Copy

If you want to add extra capabilities, such as reading certain resources (e.g., `/var/log/*`), add `CAP_DAC_READ_SEARCH` in a space-separated format as follows:

```
[Service]
AmbientCapabilities=CAP_NET_BIND_SERVICE CAP_DAC_READ_SEARCH
```

Copy

To unlock advanced file monitoring in Cribl Edge, such as scanning open files for running processes, discovering active logs automatically in Explore, and using the File Monitor Source's auto-mode feature, add `CAP_SYS_PTRACE`:

```
[Service]
AmbientCapabilities=CAP_NET_BIND_SERVICE CAP_DAC_READ_SEARCH CAP_SYS_PTRACE
```

Copy

Alternatively, you can use ACLs to Allow Cribl Edge to Read Files.

For details, see Running Edge as an Unprivileged User.

Using initd

To **enable** Cribl Edge to start at boot time with **initd**, you must run the `boot-start` command. If the user that you want to run Cribl Edge does not exist, create that user prior to executing. E.g., running Edge as user `cribl` on boot:

```
sudo $CRIBL_HOME/bin/cribl boot-start enable -m initd -u cribl
```

Copy

This will install an `init.d` script in `/etc/init.d/cribl.init.d`, and will start Cribl Edge at boot time as user `cribl`. A `-configDir` option can be used to specify where to install the script. If not specified, this location defaults to `/etc/init.d`.

If necessary, change ownership for the Cribl Edge installation:

```
sudo chown -R cribl $CRIBL_HOME
```shell

To **disable** starting at boot time, run the following command:

```shell
sudo $CRIBL_HOME/bin/cribl boot-start disable
```

Copy

To control Cribl Edge, you can use the following initd commands:

```
sudo service cribl-edge [start|stop|restart|status]
```

Copy

3.3. OS Tuning For Large Deployments

Large on-prem deployments that handle high connection volumes or data rates may require operating system (OS) tuning to maintain optimal performance. To prevent performance issues on high-load Edge Nodes, you might need to tune your OS.

Review and apply the following OS-level configurations to address common bottlenecks like connection tracking table exhaustion and buffer overruns, ensuring stable operation for your Cribl environment.



The parameters described in this section are Linux kernel and operating system settings, not Cribl Edge configuration options. They directly affect how the OS handles network connections and system resources for high-throughput Cribl Edge workloads.

Connection Tracking (Netfilter)

Applies to: Worker Nodes (handling high volumes of inbound network connections.)

In deployments where you expect a high number of inbound connections, Linux netfilter connection tracking (`conntrack`) could become a bottleneck. Connection tracking is enabled by default on most Linux distributions and maintains a table of active network flows. When this table fills up, the kernel drops packets, which can cause connection failures and data loss.

When to Tune

Monitor your system logs (`/var/log/messages` or `journalctl \-k`) for messages indicating conntrack table exhaustion:

```
nf_conntrack: table full, dropping packet
```

If you observe these messages, you should increase the `conntrack` table size and potentially adjust related parameters.

Key `Conntrack` Parameters

These `sysctl` parameters govern connection tracking behavior:

- `net.netfilter.nf_conntrack_max`: Maximum number of flows tracked concurrently. By default, it derives from the buckets value: `nf_conntrack_max= nf_conntrack_buckets × 4.`

- `net.netfilter.nf_conntrack_buckets`: Hash buckets for the `conntrack` table. If not set at module load, the kernel computes a default from total system memory (roughly `total memory in bytes / 16384`) with floor and ceiling limits applied.

For the full list of netfilter `conntrack sysctls` and their defaults, see Netfilter Conntrack Sysfs variables.

Check Current Values and Live Load

Use the following commands to inspect your current conntrack configuration and active connection count:

```
# Current max entries
sysctl net.netfilter.nf_conntrack_max

# Current active tracked flows (count)
sysctl net.netfilter.nf_conntrack_count

# Alternatively via procfs
cat /proc/sys/net/netfilter/nf_conntrack_max
cat /proc/sys/net/netfilter/nf_conntrack_count
```

Copy

Compare the count against the max. If the count approaches or equals the max during normal operation, you need to increase capacity.

Method 1 (Recommended): Increase the `conntrack` Table Size

This is the standard and generally recommended approach to eliminate `conntrack` bottlenecks by simply giving the table more capacity.

To ensure stability, set the maximum number of tracked flows high enough to maintain adequate margin above your peak observed load. For very large deployments handling hundreds of thousands of concurrent connections, you may need values of 4 million or higher.

```
# For example, to set approximately 2 million tracked connections (adjust per sizing)
echo "net.netfilter.nf_conntrack_max=2097152" >> /etc/sysctl.conf
sysctl -p
```

Copy

Adjust the value based on your deployment's actual connection load.

If you substantially increase `nf_conntrack_max`, review `nf_conntrack_buckets` for balance. By default, the kernel derives buckets from available memory, and max defaults to approximately 4 times the bucket count. Maintaining this ratio helps ensure efficient hash table performance.

Method 2 (Extreme Case): Disable Connection Tracking

In extremely high-throughput, purpose-built environments where Cribl Stream is the only service and network overhead is a top concern, disabling conntrack entirely can be considered an aggressive measure. While this is often the simplest and most effective measure for some on-prem deployments, approach it with caution due to its side effects.

How to Disable conntrack (Persistent)

To prevent the module from loading at boot:

```
echo "install nf_conntrack /bin/true" >> /etc/modprobe.d/disable-conntrack.conf
```

Copy

Disabling conntrack is **NOT generally recommended** and carries significant consequences:

- Breaks stateful firewalls: Most modern firewall rules (iptables, nftables) rely on conntrack for stateful filtering. Disabling it will break stateful firewalls, requiring a complete rewrite of your rules to be stateless.
- Breaks NAT/Kubernetes: Network Address Translation (NAT) and IP masquerading are heavily dependent on conntrack. Disabling it will break networking for container technologies like Kubernetes and Docker.

Kubernetes Deployments

If you run Cribl Edge on Kubernetes, **do not disable netfilter or conntrack**. Kubernetes networking (including kube-proxy and network policies) depends on conntrack for proper operation. Tune conntrack parameters as described in Method 1 instead.

File Descriptor Limits

Applies to: Leader and Worker Nodes

The Cribl Edge systemd unit sets `LimitNOFILE=65536` by default, which aligns with our minimum recommendation. This limit applies per process (not systemwide or per user), meaning the API and each Worker Process can each open up to `65,536` file handles. This is generally more than sufficient for high-volume workloads, including those involving:

- File-based Destinations (S3, Filesystem, etc.), especially with partition expressions that possess a high cardinality. For details, see Amazon S3 Better Practices.
- Large numbers of concurrent inbound and outbound connections

For more information on running Cribl Stream as a service, see Run Cribl Stream.

Diagnosing `Too Many Open Files` Error

The default value is typically sufficient for normal operation. Increasing the limit is generally not the correct fix; it usually only postpones the necessity of identifying and resolving the root cause of the process hitting the resource ceiling.

Start your investigation by determining which process is affected. The `cribl.log` file will reflect a `too many open files` error, referencing one of two error codes:

- `EMFILE (Too many open files)`: This means the per-process limit has been reached. This is the common case that the LimitNOFILE setting addresses.
- `ENFILE (File table overflow)`: This means the systemwide (kernel-level) limit for all open files has been reached. The fix for this (increasing the `fs.file-max sysctl`) is outside the scope of this section, as it's rarely encountered.

The scenario that leads to the common `EMFILE` error is usually the API Process being overwhelmed due to Worker Processes being too busy. The API Process always receives the connection attempt first, then hands the file descriptor off to a Worker Process.

The `EMFILE` condition typically occurs when a large number of clients (on the order of thousands) establish connections that are not closed have excessively long timeouts. Since Worker Processes are often unable to accept the handoff of these file descriptors from the API, the API Process accumulates thousands of connections waiting to be serviced. This accumulation of active or stuck connections rapidly exhausts the file descriptor limit for the API Process. The proper fix, therefore, is to address why the Worker Processes are too busy.

Start by analyzing your Pipelines using the Pipeline Profiling feature to identify bottlenecks and reduce Worker load. For details, see Pipeline Profiling.

How to Increase the Per-Process Open File Limit (if Necessary)

If, after investigation, you've confirmed that the workload legitimately requires more file handles (such as, in extremely large, high-volume, or unique deployments), you can increase this per process limit via a systemd override:

```
# Create or edit the systemd override
sudo systemctl edit cribl
```

Copy

Add the following (example) content:

```
[Service]
LimitNOFILE=262144
```

Copy

Save the file and restart the Cribl Stream service for the change to take effect:

```
sudo systemctl restart cribl
```

Copy

You can verify the current limit for running processes by checking `/proc/<pid\>/` limits for any Cribl Edge process.

```
# For exampalefor PID 7284
cat /proc/7284/limits
```

Copy

UDP Receive Buffers

Applies to: Worker and Edge Nodes (Nodes hosting UDP input Sources).

For high-rate UDP ingest (such as Syslog, NetFlow, or IPFIX), the default kernel UDP receive buffers may be insufficient, leading to packet drops before data reaches Cribl Stream. Increasing these buffers can significantly reduce packet loss.

When to Tune

Cribl recommends tuning UDP buffers to accommodate short-lived but large data spikes. Tuning UDP buffers is recommended to accommodate short-lived but large data spikes.

Before tuning the buffer size, you should confirm that the data is being properly distributed across all the Worker Nodes in the Group. If the data is improperly load-balanced and only hitting a subset of your Nodes, you must address the distribution issue first. Fixing the load balancing might resolve the packet drop problem entirely, making buffer adjustments unnecessary.

For permanent solutions to consistently high data volume, scaling your environment (adding more worker processes or nodes) is the correct approach.

Monitor for UDP packet drops using:

```
# On modern operating systems, check for packet receive errors using ss
ss -ua
# Alternatively, if ss is unavailable
netstat -su
```

Copy

Look for increases in `packet receive errors` or `RcvbufErrors` under the UDP section.

Method 1: Kernel-Level Tuning (System-Wide OS Configuration)

This method applies the buffer increase to **all** UDP sockets on the system.

Increase the kernel's UDP receive buffer sizes:

```
# Set to 25 MB (adjust based on your volume)
echo "net.core.rmem_max=26214400" >> /etc/sysctl.conf
echo "net.core.rmem_default=26214400" >> /etc/sysctl.conf
# Apply the changes immediately without rebooting:
sysctl -p
```

Copy

Method 2: Per-Input via Cribl Stream Configuration

We generally recommend configuring the buffer size within Cribl Stream (on the specific UDP Source) for two main benefits:

- **Portability:** The setting becomes part of your Cribl Stream configuration, ensuring it is automatically applied to newly deployed Nodes without requiring manual OS tuning on each one.
- **Scope:** The larger buffer is applied only to the specific UDP sockets used by Stream Sources that have this setting defined, avoiding the general application to all system sockets.

Start with `25` MB and adjust based on your data rates and observed drop rates.

For detailed UDP tuning guidance specific to Cribl Stream, see:

- Syslog Source - UDP Tuning
- UDP (Raw) Source

SELinux Configuration

Applies to: Leader, Worker, and Edge Nodes.

In hardened environments running SELinux in enforcing mode, ensure proper SELinux contexts are set on Cribl directories to allow Stream to operate correctly. Use SELinux audit logs to troubleshoot any denials and adjust policies as needed.

For detailed guidance on running Cribl Stream with SELinux, see SELinux Configuration.

3.4. Run Edge As An Unprivileged Linux User

Run Cribl Edge as an unprivileged (non-root) Linux user.

Privileged access might be necessary if Cribl Edge needs to read certain resources (e.g., `/var/log/*`), or to listen on low ports 1-1024. Features like auto-discovery of logs and information in the Processes UI also require permissions to access `/proc`. The regular non-root permissions are not sufficient in these cases.

There are two alternatives to running Cribl Edge as `root`:

- Set Linux capabilities that grant Cribl Edge sufficient rights to perform specific privileged tasks. For details, see Set Capabilities for Cribl Edge below.
- Take advantage of Linux systems' option to layer an Access Control List (ACL) over the default Linux permissions. By using ACLs, you can assign a more specific set of permissions to a file or directory without (necessarily) changing the base ownership. For details, see Using ACLs to Allow Cribl Edge to Read Files.

Set Capabilities for Cribl Edge

Capabilities are permissions that grant privileged processes sufficient rights to accomplish a specific task, based on a kernel privilege. To run Cribl Edge as non-root user, consider setting the following capabilities:

Capability	Permissions
`CAP_NET_ADMIN`	Allows Cribl Edge to identify network listeners on the node.
`CAP_NET_BIND_SERVICE`	Allows Cribl Edge to push Sources that bind to TCP/UDP port numbers below `1024`.
`CAP_DAC_READ_SEARCH`	Allows the `cribl` user to access files in **Explore** > **Files** > Manual/Browse, and to access the File Monitor Source's Manual mode feature. This capability bypasses the default Linux permissions for files and directories.
`CAP_SYS_PTRACE`	Allows the `cribl` user to scan open files for running processes, to discover active logs in **Explore** > **Files** > Auto, and to access the File Monitor Source's Auto-mode feature.

For details about setting these capabilities, see Persisting Overrides.

Making Docker Containers Visible to Edge

You can enable Cribl Edge to access statistics and information about any running container on its host. To do this, add the `cribl` user to the `docker` group, then restart Cribl Edge.

OS-Specific Options

To read file resources on a Linux system that are typically restricted to the root user, you can add the `CAP_DAC_READ_SEARCH` capability. For example:

On some OS versions (such as CentOS), you must add an `-i` switch to the `setcap` command. For example: `# setcap -i cap_dac_read_search=+ep $CRIBL_HOME/bin/cribl`

Upgrading Edge will remove the `CAP_DAC_READ_SEARCH` capability from the `cribl` executable, so you'll need to re-run the appropriate `setcap` command after each upgrade.

Fallbacks from Privileged Access

If installing and running Edge with `root`-level privileges is forbidden or impractical in your environment, certain Sources, like Exec and System Metrics, can run on a user with lower permissions. You can also run the File Monitor Source in `Manual` mode, and collect from any files that this user can read.

3.5. Install Edge On Linux Via RPM

Install Edge in a tightly managed Linux environment by using the signed RPM package installation method.

The RPM will create a `cribl-edge` user and a `cribl-edge` group. The `cribl-edge` user will own the `cribl-edge` configuration file (stored in the `/etc/sysconfig` directory). The `config` file contains the Leader's auth token. To keep the Leader's auth token secure, only `root` and the `cribl-edge` user can read/modify the `config` file contents.

Download Cribl Edge as an RPM Package

To download Cribl Edge as a signed RPM package:

1. Go to cribl.io/download.
2. Select the dropdown in the Software area labeled **Select Release Version**.
3. Choose the Cribl Edge RPM version and type: `<version> x64 Cribl Edge RPM`, or `<version> ARM Cribl Edge RPM`.
4. Select **Download now**.

Verifying the RPM Package Signature

To verify the RPM package signature, you'll need to enable local GPG package checking, then download the GPG key, import it, and verify its signature. We'll go over how to accomplish both of these steps below.

Enable Local GPG Package Checking

Your environment may not have the GPG check for local packages enabled by default.

To enable the local package GPG checks for `yum` or `dnf`, add `localpkg_gpgceheck=1` to the `main` stanza in:

- `/etc/yum/conf` for `yum`
- `/etc/dnf/dnf.conf` for `dnf`

For example:

```
[main]
...
localpkg_gpgcheck=1
```

Copy

Download and Import the GPG Key

Cribl's public GPG key ensures the authenticity and integrity of the Cribl RPM package.

1. Import the public GPG key to `rpm` using this command:

   ```
   rpm --import https://cdn.cribl.io/dl/CRIBL_RPM_PUBLIC_GPG
   ```

   Copy

2. Verify the package signature using this command:

   ```
   rpm -Kv <rpm-file>
   ```

   Copy

   This command ensures the package's integrity and authenticity, and returns information about the package. It also checks if anyone has tampered with the package since the original developer or distributor signed it. This helps prevent you from installing potentially malicious or compromised packages.

Installing Edge via RPM

Follow the installation steps to install the RPM package on a Cribl Linux Edge Node.



Out of the box, the RPM will install the software into the `/opt` directory. The RPM package will configure but not enable the Systemd service. You will need to enable the software service separately.

To install Cribl Edge via RPM:

1. Open a CLI and run the following command, drawing the file name from the cribl.io/download page:

   ```
   sudo yum install https://cdn.cribl.io/dl/<version>/<file name>
   ```

Copy

You can quickly get the CDN file name from the checksum links for each download.

This command does the following:

- Creates the `cribl-edge` user and group, if it doesn't exist.
- Puts the contents of Cribl's TGZ into `/opt/cribl-edge` and makes `root/root` the owner.
- Creates the `cribl-edge` systemd service, which is disabled by default.
- Creates `/etc/sysconfig/cribl-edge.conf` owned by `cribl-edge:cribl-edge` and only accessible by the `cribl-edge` user.
- Creates `/var/lib/cribl-edge/` owned by `cribl-edge:cribl-edge` and only accessible by the `cribl-edge` user.
- Sets `CRIBL_INSTALL_TYPE` to `RPM` in the service destination, to block distributed upgrades, as upgrading from the Leader bypasses the security of RPM.
- Sets `CRIBL_VOLUME_DIR` to `/var/lib/cribl-edge` in the service destination, so configs and logs are saved to the `cribl-edge` directory.

2. Edit the environment variables in `/etc/sysconfig/cribl.conf`, in order for Cribl Edge to connect to a Leader. The RPM installs `cribl-edge` in standalone mode by default (`CRIBL_DIST_MODE = edge`).

   To deploy Edge in a managed deployment, edit these environment variables:

   - `CRIBL_DIST_MODE = managed-edge`
   - `CRIBL_DIST_LEADER_URL=tls://<authToken>@leader:4200`

3. Enable optional Linux capabilities.
   `CAP_DAC_READ_SEARCH` lets you read files without ownership/permission checks (such as `/var/log/nginx` or application logs under service accounts), and `CAP_SYS_PTRACE` lets you trace processes and read `/proc/<pid>/` data.
   To enable them, add this configuration key to the systemd service file, located at `/etc/systemd/system/cribl-edge.service` or `/lib/systemd/system/cribl-edge.service`:

   ```
   [Service]
   AmbientCapabilities=CAP_DAC_READ_SEARCH CAP_SYS_PTRACE
   ```

Copy

4. Enable the service:

- `sudo systemctl enable cribl-edge` to enable the service on boot.

- `sudo systemctl start cribl-edge` to start it manually.

Upgrading Edge via RPM

See Upgrade RPM Installations.

Uninstalling Edge via RPM

To remove Cribl that you installed via RPM, use the command line:

```
sudo yum remove cribl-edge
```

Copy

Uninstalling Cribl removes the `cribl` or `cribl-edge` service, the `cribl` binary from `/opt/`, and the service file from `/lib/systemd/system`.

Configuration files are maintained, along with any environment variables you edited when you first installed Cribl. This means you can pull in your previous configuration(s) if you decide to reinstall Cribl via RPM.

The `cribl-edge` directory is also maintained, along with the data, logs, and defaults of your configuration.

3.6. Install Edge On Linux Via .Deb

Install Cribl Edge in a Linux environment using a .deb package.

The installation conforms to restrictions for Linux installations, such as system requirements.

Install Cribl Edge from a .deb Package

Follow these steps to install an Edge Node on Linux from a .deb package:

1. Download the Cribl Edge .deb package. Go to cribl.io/download and select the dropdown in the Software area labeled **Select Release Version**.
2. Choose the Cribl Edge .deb version and type: `<version> x64 Cribl Edge DEB`, or `<version> ARM Cribl Edge DEB`.
3. Select **Download now**.
4. Open a CLI and run the following command to install the package with the apt utility:

```
sudo apt install <path-to-deb-package>
```

Copy

   This command does the following:

   - Creates the `cribl-edge` user and group, if it doesn't exist.
   - Puts the contents of the Cribl package into `/opt/cribl-edge` and makes `root/root` the owner.
   - Creates the `cribl-edge` systemd service, which is disabled by default.
   - Creates `/etc/default/cribl-edge` owned by `cribl-edge:cribl-edge` and only accessible by the `cribl-edge` user.
   - Creates `/var/lib/cribl-edge/` owned by `cribl-edge:cribl-edge` and only accessible by the `cribl-edge` user.
   - Adds the Cribl Edge service definition file at `/lib/systemd/system/cribl-edge.service`.
   - Sets `CRIBL_INSTALL_TYPE` to `DEB` in the service definition, to block distributed upgrades, as upgrading from the Leader bypasses the security of a .deb package.
   - Sets `CRIBL_VOLUME_DIR` to `/var/lib/cribl-edge` in the service definition, so configs and logs are saved to the `cribl-edge` directory.

5. Edit the environment variables in `/etc/default/cribl-edge`, so Cribl Edge can connect to a Leader. The .deb package installs `cribl-edge` in standalone mode by default (`CRIBL_DIST_MODE = edge`).

   To deploy Edge in a managed deployment, edit these environment variables:

   - `CRIBL_DIST_MODE=managed-edge`
   - `CRIBL_DIST_LEADER_URL=tls://<authToken>@leader:4200`

6. During installation, the package configures but does not enable the systemd service. Enable the service:
   - `sudo systemctl enable cribl-edge` to enable the service to start on boot.
   - `sudo systemctl start cribl-edge` to start it manually.

To ensure that the `cribl status` command can run correctly when running via systemctl, set the `CRIBL_VOLUME_DIR` environment variable:

```
sudo bash
export CRIBL_VOLUME_DIR=/var/lib/cribl
```

Copy

Upgrade a .deb Installation

See Upgrade .deb Installations.

Uninstall a Cribl Edge .deb Package

To remove Cribl Edge that you installed from a .deb package, run:

```
sudo apt remove <path-to-deb-package>
```

Copy

Uninstalling removes the `cribl-edge` service, the `cribl` binary from `/opt/`, and the service file from `/lib/systemd/system`.

Configuration files are maintained, along with any environment variables you edited when you first installed Cribl. This means you can pull in your previous configuration(s) if you decide to reinstall Cribl via a .deb package.

The `cribl-edge` directory is also maintained, along with the data, logs, and defaults of your configuration.

3.7. FIPS Mode For Cribl Edge (Linux)

Federal Information Processing Standards FIPS represent a set of US government standards and guidelines for information security. You can deploy Cribl Edge in **FIPS mode**. This mainly restricts the cryptographic algorithms used within Cribl Edge and also enforces FIPS compliant password requirements.



Do not start Cribl Edge without FIPS mode enabled, or else you will be unable to run it in FIPS mode later. You must enable FIPS mode as described in this section, after installing **but before starting** Cribl Edge.

Requirements

This section describes the system and password requirements for FIPS mode.

FIPS Mode System Requirements

To run Cribl Edge in FIPS mode, your environment must satisfy the following requirements.

Operating Systems

Must support FIPS 140-2 or FIPS 140-3. Several Linux distributions meet this standard. See NIST's list of tested configurations here.



While RHEL 9 can be configured to run in FIPS mode at the operating system level, this **does not automatically enable FIPS mode within Cribl Edge.** You must still follow the procedures outlined in this document to explicitly enable Cribl Edge's FIPS mode. Cribl Edge relies on its own configuration to ensure FIPS compliance, independent of the OS-level FIPS setting.

OpenSSL

The operating system environment must have a FIPS validated version of OpenSSL installed.

- **For Cribl Edge versions 4.5.0 through 4.8.1**: The FIPS provider version must be 3.0.8 or 3.0.9, and the system-wide OpenSSL version must also be 3.0.8 or version 3.0.9, with its associated certificate.
- **For Cribl Edge starting with version 4.8.2**: The FIPS provider version requirement must be >= 3.0.5, and the system-wide OpenSSL version is no longer a factor. Cribl Edge now verifies the OpenSSL version bundled within the binary.

Enterprise Linux 9 is the only version of Enterprise Linux that includes OpenSSL version 3.0.9 out of the box. While it's possible to manually install compatible versions of OpenSSL on older Enterprise Linux versions, this process is complex and generally not recommended. Therefore, the minimum recommended Enterprise Linux version for running Cribl Edge in FIPS mode is 9.

Even when running Cribl Edge on a FIPS-enabled RHEL 9 system, you must still explicitly enable Cribl Edge's FIPS mode as described in the Running Cribl Edge in FIPS Mode section below. The OS being in FIPS mode is not sufficient for Cribl Edge to operate in a FIPS-140 compliant manner.

FIPS Mode Restrictions on Cryptographic Algorithms

When running in FIPS mode, Cribl Edge uses only those cryptographic algorithms that satisfy the FIPS standards. This means that Cribl Edge operates exclusively with FIPS-compliant cryptographic algorithms, disallowing the use of non-compliant algorithms like MD5 (message-digest) or CRC-32 (Cyclic Redundancy Check 32) algorithms.

To ensure proper functionality in FIPS mode, the FIPS provider version must be at least `3.0.5`.

Therefore, in FIPS mode:

Cribl expressions **that rely on MD5 or CRC-32 will fail silently.** This includes expressions that use the `C.Mask.md5()` method and can in turn affect the behavior of parent Functions and their existing parent Pipelines.

The UI will hide certain options normally made available by the typeahead feature.

No Source, Destination, or Email Notification target that uses the Amazon AWS SDK v2 will perform checksums. This is because that SDK uses MD5 to verify checksums, and it applies to both incoming or outgoing payloads. Specifically:

- The Amazon Kinesis Data Streams Source will not perform checksums and the **Verify KPL checksums** setting will not be available.
- The Amazon SNS Notification target will not perform checksums.
- The following Destinations will not perform checksums:
    - Amazon CloudWatch Logs
    - Amazon Kinesis Data Streams
    - Amazon SQS
    - Google Cloud Storage
    - MinIO
    - Prometheus

Role-Based Access Control (RBAC) Controls

Cribl Edge's FIPS mode requires RBAC to be enabled for enhanced security. This ensures proper access management within FIPS-compliant environments. To enable RBAC, the Cribl Edge instance has to be running in Distributed mode. FIPS mode is available on certain license tiers. For details, see Pricing.

If FIPS mode is already enabled and RBAC is subsequently disabled, Cribl Edge will automatically disable FIPS mode and log a message stating:

```
"Disabling FIPS due to Role-based Access Control (RBAC) being disabled."
```

Check `<cribl_install_dir>/local/_system/instance.yml` to ensure Cribl is running in Distributed mode. The default installation directory is typically `/opt/cribl`. The `mode` setting in this file should be set to `distributed`. If it is set to `standalone`, you will need to switch to a Distributed deployment to use FIPS mode with RBAC enabled.

FIPS Mode Password Rules

When Cribl Edge is in FIPS mode, all passwords must:

- Contain eight or more characters.
- Use characters from three or more of the following categories:
    - Lowercase letters.
    - Uppercase letters located after the first character in the password.
    - Digits located before the last character in the password.
    - Non-alphanumeric ASCII characters such as `#`, `!`, or `?`.
    - Non-ASCII characters such as `ñ`, `€`, or emoji.

As of Cribl Edge version 4.5.0, these rules apply for all passwords, whether or not Cribl Edge is running in FIPS mode, with one exception:

- For Cribl Edge not running in FIPS mode, local users whose passwords existed before Cribl Edge 4.4.4 was released can continue to use their passwords, even if those passwords do not satisfy the rules.
- When these users change to new passwords, the new passwords must satisfy the rules.
- New users must create passwords that satisfy the rules.



If you get locked out of your account, you need to reset your password manually.

Running Cribl Edge in FIPS Mode

To begin using Cribl Edge with FIPS enabled, complete either the non-systemd or the systemd procedure below, **before you start Cribl Edge for the first time after installing**.

Enabling FIPS Mode on a Non-systemd Distribution

If the Linux distribution on which you run Cribl Edge **does not** use systemd:

1. Complete **either** of the two following steps:
   - Set the `CRIBL_FIPS` environment variable to `1` (true), **or**,
   - Edit `cribl.yml`, adding this top-level element:

     ```
     fips: true
     ```

     Copy

     Note that `fips: true` is the entire top-level element. For this element, the following "stanza" syntax would be incorrect - **do not** use it:

     ```
     # stanza syntax is incorrect for the fips element
     # even if other elements use it
     fips
       fips: true
     ```

     Copy

2. If you have installed a non-default OpenSSL binary (otherwise, skip this step):
   - Run the following command, and in the output, note the values of `MODULESDIR` and `OPENSSLDIR`.

     ```
     <binary_of_non-default_OpenSSL> version -a
     ```

     Copy

   - Set `OPENSSL_MODULES` to the value of `MODULESDIR`.
   - Set `CRIBL_OPENSSL_DIR` to the value of `OPENSSLDIR`.
3. Ensure RBAC is configured in Cribl Edge.

4. Start Cribl Edge.

Finally, verify that you are in FIPS mode as explained below.

Enabling FIPS Mode on a systemd Distribution

If the Linux distribution on which you run Cribl Edge **does** use systemd:

1. Run the following command, and in the output, note the values of `MODULESDIR` and `OPENSSLDIR`. (As shown, the command will run the default OpenSSL binary. If you have installed a different OpenSSL binary, you can modify the command to run that one instead.)

   ```
   openssl version -a
   ```

   Copy

2. Run the following command, replacing the placeholders with the relevant paths from your environment. (This assumes that you do not already have a `nodejs.cnf` file. If you **do** have one, complete the workaround below instead of this step.)

   ```
   <path_to_Cribl_binary> generateFipsConf -d <path_to_OpenSSL_root_directory>
   ```

   Copy

   If you encounter issues generating the FIPS configuration file, a discrepancy in the OpenSSL path might be the cause. For more detailed troubleshooting specific to RHEL 9, refer to the knowledge base article: Installing Cribl Edge in FIPS Mode on RHEL 9.

   An example of a successful command and its output:

   ```
   useralice@123456791234:/# /opt/cribl/bin/cribl generateFipsConf -d /lib/usr/ssl
   /opt/cribl/state/nodejs.cnf
   ```

   Copy

   This command creates `nodejs.cnf` - the configuration file that NodeJS uses to run in FIPS mode as Cribl Edge starts.

3. Run the following command to open an empty temporary file in an editor:

   ```
   systemctl edit cribl
   ```

Copy

4. Add the following content to the file, replacing the first placeholder with the value of `MODULESDIR` that you noted in Step 1, and the second with the path to your `nodejs.cnf` file:

```
[Service]
Environment="OPENSSL_MODULES=<value_of_MODULESDIR>"
Environment="OPENSSL_CONF=<path_to_nodejs.cnf_file>"
```

Copy

5. Save and exit the editor.
6. Run the following commands to reload the systemctl overrides and start Cribl:

```
systemctl daemon-reload
systemctl start cribl
```

Copy

7. Finally, verify that you are in FIPS mode as explained below.

When a `nodejs.cnf` File Already Exists

If you already have a `nodejs.cnf` file, replace Step 2 above with the following workaround:

1. Create a file with the following content, replacing `<placeholder>` with the value of `OPENSSLDIR`. Name the file something other than `nodejs.cnf`.

```
nodejs_conf = nodejs_init
openssl_conf = nodejs_init

.include /<placeholder>/fipsmodule.cnf

[nodejs_init]
providers = provider_sect

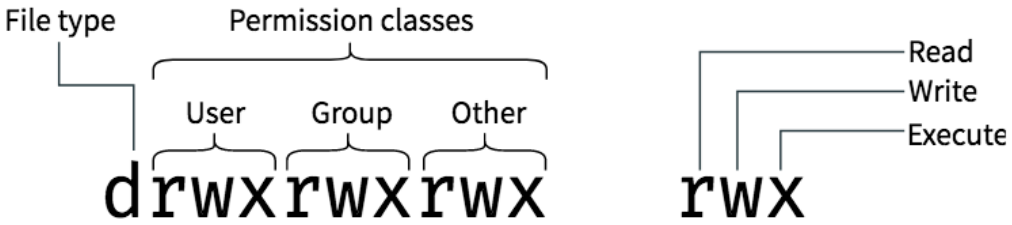
[provider_sect]
default = default_sect
# The fips section name should match the section name inside the
# included fipsmodule.cnf.
fips = fips_sect
[default_sect]
activate = 1
```

Copy

2. Resume the main procedure above, starting with Step 3.

3.8. Use ACLs To Allow Cribl Edge To Read Files

Running Cribl Edge as an unprivileged user is a best practice. However, without modifying the default Linux permissions, you will run into issues in accessing files owned by other users.



Default Linux permissions

Linux systems allow you to layer an Access Control List (ACL) on top of the default Linux permission set. With ACLs, you can apply a more specific set of permissions to a file or directory without (necessarily) changing the base ownership and permissions. For details, see Introduction to ACLs.

As an example, you might want to read data from the `/var/log` directory. This directory is typically owned by the `root` user, with a permission set of `750` on the directory. This means the `cribl` user will not be able to read or list the files in the directory, because the `Other` group has zero permissions.

To achieve compliance with benchmarks such as CIS or NIST, we can use the ACLs to grant the `cribl` group access to this folder and any files, without disturbing the current permissions.

> **CIS Benchmark 4.2.3**
>
> Make sure that permissions are configured on all log files. Log files must have the correct permissions to ensure that sensitive data is archived and protected. `Other/world` should not have the ability to view this information. `Group` should not have the ability to modify this information.

To accomplish this, we can grant the `cribl` group `read` and `execute` access to the files and directories inside `/var/log`, by running this command:

```
setfacl -Rm g:cribl:r-X /var/log
```

Copy

Breaking down the command's options':

- `-R` : Recursive

- `-m` : Modify
- `g:cribl` = `cribl` group, this could be `u:cribl` if you wanted to limit to the `cribl` user.
- `r-X`: Read and execute. Capital `X` means execute only on directories.

This modifies only the current files in the directory, if you want the appropriate ACL applied to any future files created here, add the `-d` flag (for default):

```
setfacl -Rdm g:cribl:r-X /var/log
```

Copy

Now, any rotated or created files will apply the ACL set.

Checking the ACLs

To verify the ACLs on a file or directory, run the following command:

```
getfacl <file or folder>
```

Copy

This will output a listing of the applied ACLs, including the directory's defaults:

```
$ getfacl /<directory>
# file: <file>
# owner: <owner>
# group: <group>
user::rwx
group::rwx
other::---
default:user::rwx
default:user:<user>:rwx
default:group::rwx
default:mask::rwx
default:other::---
```

Copy

Installing ACL Utilities

The ACL utilities might not be installed, by default, on the OS. For example, on Ubuntu (Debian-based) systems, you will need to install the `acl` package. For Debian-based tools using apt:

```
apt install acl
```

Copy

For Red Hat-based tools using yum:

```
yum install acl
```

Copy

4. Cribl Edge On Windows

Check the options and limitations for running Cribl Edge on Windows.

Cribl Edge on Windows offers easy-to-use tools for exploring and collecting Windows events. You can run Cribl Edge on a supported Windows environment and collect events via the Windows Events API.

Limitations

Cribl Edge on Windows is currently subject to the following limitations:

Modes

Cribl Edge on Windows supports only the following modes:

- `Edge: Single`
- `Edge: Managed Edge (managed by Leader)`

This means you can't switch Cribl Edge on Windows into Cribl Stream mode (Single-instance, Worker, or Leader).

You can, however, switch between the Cribl Edge supported modes via the UI, in **Settings** > **Distributed Settings** > **Mode**.



Do not select an unsupported mode from this drop-down! Doing so will cause the Cribl service to fail.

Sources and Destinations

Cribl Edge on Windows supports the same Sources and Destinations as Cribl Edge on Linux, with the following exceptions:

- The Kubernetes Logs and Kubernetes Metrics Sources. Cribl Edge on Windows does not support Kubernetes deployments.
- The Kafka Destination does not support Kerberos authentication on Windows.

Sources on Windows Only

The following Sources are available **only** when running Cribl Edge on Windows (not on Linux):

- Windows Event Logs
- Window Metrics

Functions

The Grok Function is unavailable on Windows. If you include it in a Pipeline, Cribl Edge processing will skip over it.

Data Formats

Cribl Edge on Windows does not support reading or writing Parquet files. This is a limitation on the following Destinations: Amazon S3, Azure Blob Storage, MinIO, and FileSystem/NFS.

4.1. Install Cribl Edge On Windows

Install and deploy Cribl Edge Nodes on-prem on Windows.

You can install Cribl Edge in a supported Windows environment.

Minimum Requirements

The following requirements are the minimum for installing and running an Edge Node on Windows.



Below is a summary of requirements for a Cribl Edge deployment. See Requirements for more detailed information.

OS Requirements

Cribl Edge supports these versions of Windows:

Version	Notes
Windows 10 and 11	You can deploy Cribl Edge on laptops, desktops, and workstations running Windows 10 and 11. Keep these things in mind when deploying Cribl Edge on Windows 10 and 11 workstations: • Cribl Edge might lose some data sent to TCP Destinations after a workstation resumes from a suspended or sleep state. This happens when the data is in local network buffers but hasn't been sent out to the Destination. Instead, you can send data to an HTTP Destination. • You might see partially broken events after a laptop resumes from a suspended state. • HTTP Destinations might contain duplicate events. • Some users have observed excessive logging when a Leader is unavailable to receive heartbeat messages from an Edge Node.
Windows Server 2016 Windows Server 2019 Windows Server 2022 Windows Server 2025	Functionality is the same as generally supported Windows Server platforms.

Application Requirements

Minimum application requirements for Edge Nodes are:

- ~1 Ghz processor
- 1024 MB RAM
- 5 GB free disk space per volume for operational purposes

Before You Install Cribl Edge on Windows

1. Set a compatible browser as your default browser: Microsoft Edge, Firefox, or Chrome. (We support the five most-recent versions of these browsers. Cribl Edge is not compatible with Internet Explorer.)
2. Ensure that the required ports are available (see Ports).
3. Go to the Cribl Download page and set the **Software** drop-down to `Cribl Edge for Windows`.
4. Click **Download Now** to get the `.msi` installer.

You can also install a Windows Edge Node in a Distributed deployment by running the bootstrap script in your command prompt.



To install Cribl Edge, you might need to add exceptions to your antivirus or endpoint security tools (such as Windows Defender). For details, see Stop Endpoint Security from Blocking Edge on Windows.

About Environment Variables

Environment variables specified during installation are added to the service configuration in the Windows registry under `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\cribl` in the `Environment` key.

When setting environment variables:

- You can add but not remove environment variables using the interactive wizard.
- Variables are merged in the following priority order (highest to lowest):
    1. Variables specified via command line during installation.
    2. Existing environment variables already configured for the service.

This ensures that command-line specifications take precedence over existing configurations, while preserving any previously set variables not explicitly overridden.

The `CRIBL_VOLUME_DIR` variable is stored separately in the Windows registry using the NSSM configuration path, which is different from where other environment variables are stored. NSSM is the service management tool used by Cribl Edge to run as a Windows service. For details, see What is the `nssm.exe` process in the Frequently Asked Questions section below.

Select the Installation Type

You now have two installation options:

- Launch the `.msi` to install via an interactive wizard.
- Use the `.msi` to install via a command prompt, or to script bulk installs.

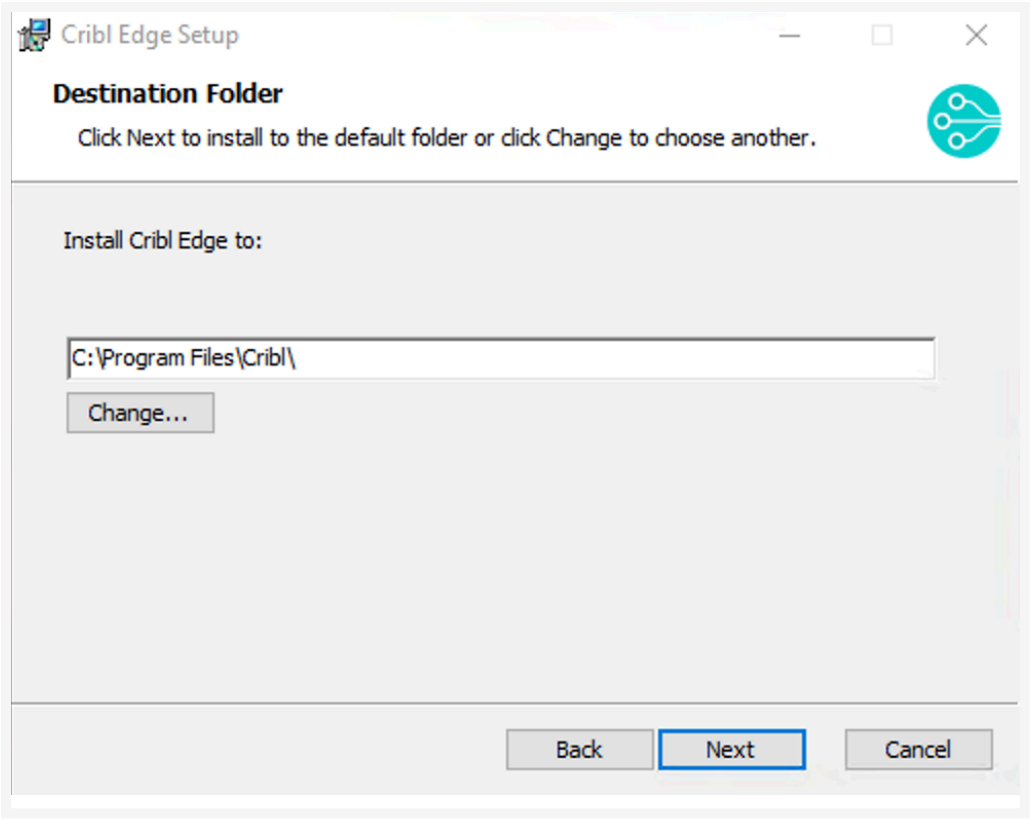
Either method installs Cribl Edge as a Windows service. This enables Cribl Edge to automatically restart whenever the Windows Server reboots.



Please see Known Issues for any current limitations on automatic version upgrades.

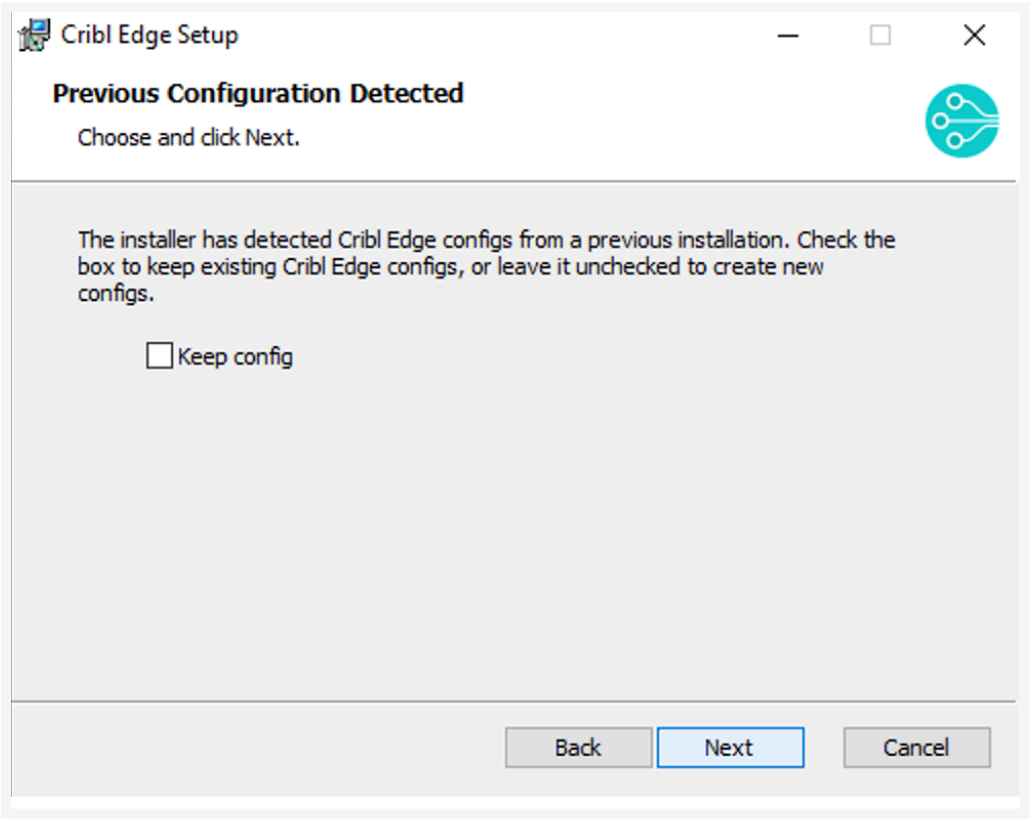
Use the Wizard

1. Double-click the Cribl Edge `.msi` to launch the Cribl Edge Setup Wizard. Select **Next** to start.
2. Read the **End-User License Agreement** and check the box to accept the terms. For the most current copy and details, see Terms of Service.
3. On the wizard's next page, confirm or change the **Destination Folder**. (The default path will expand as `C:\Program Files\Cribl`).



Select location

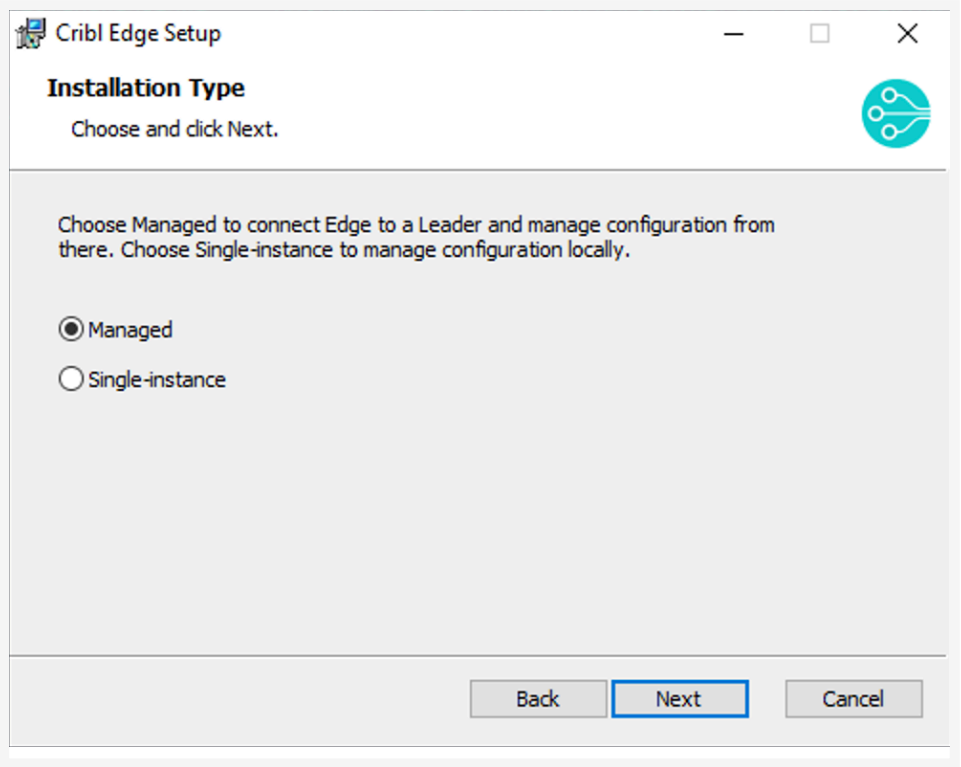
4. If the installer detects a config file from a previous Cribl Edge installation, the modal will ask you if you want to keep your current config or create a new config file for the installation. Select **Keep config** to persist your previous configurations stored in the `instance.yml` file.



Keep Config

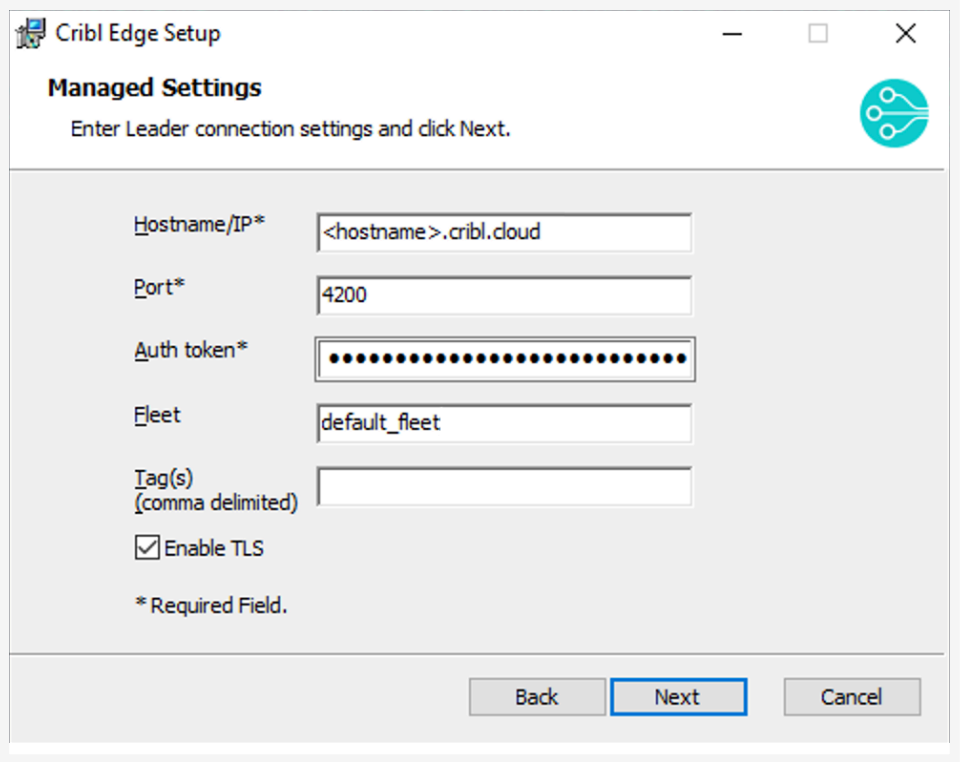
5. Select either **Managed** or **Single-instance** as the installation type. See Types of Deployment for information on these options.
   If you chose **Keep config** in the previous window, the installer will skip the installation type selection.

   For Cribl.Cloud, Select **Managed Edge Node**.



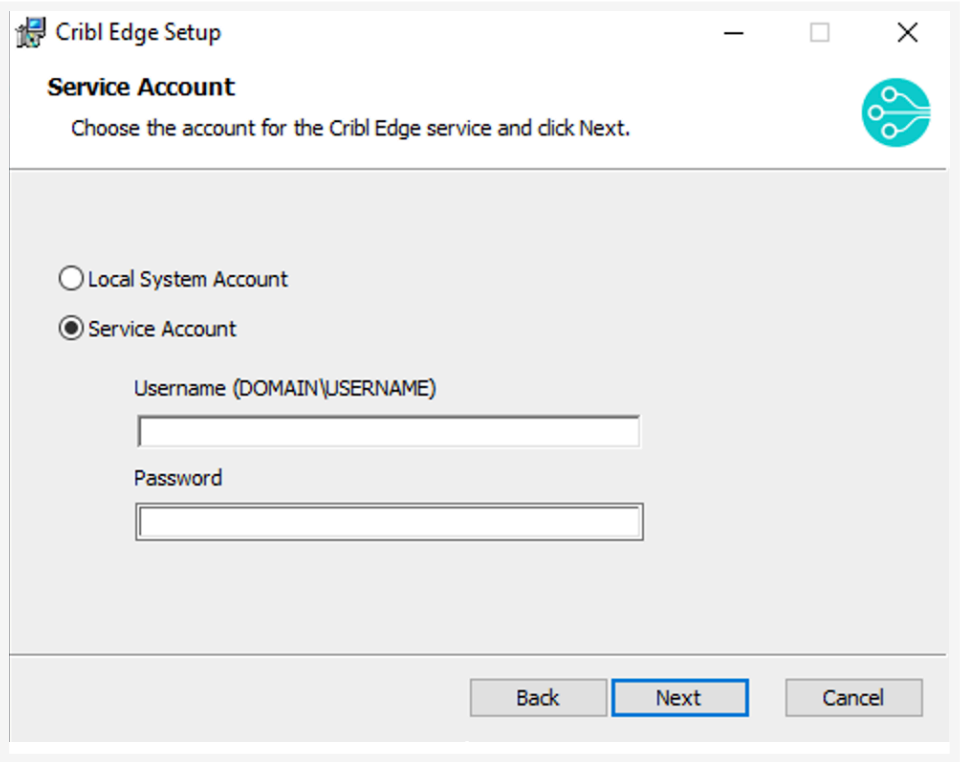
Select installation type

6. If you select **Managed** mode, enter the Leader hostname or IP address, port, auth token, and optionally the Fleet and any tags. For details on getting these credentials from Cribl.Cloud, see Retrieve Cribl.Cloud Credentials.
   For sites using secure TLS connections to the Leader, including Cribl.Cloud, ensure that **Enable TLS** is selected.

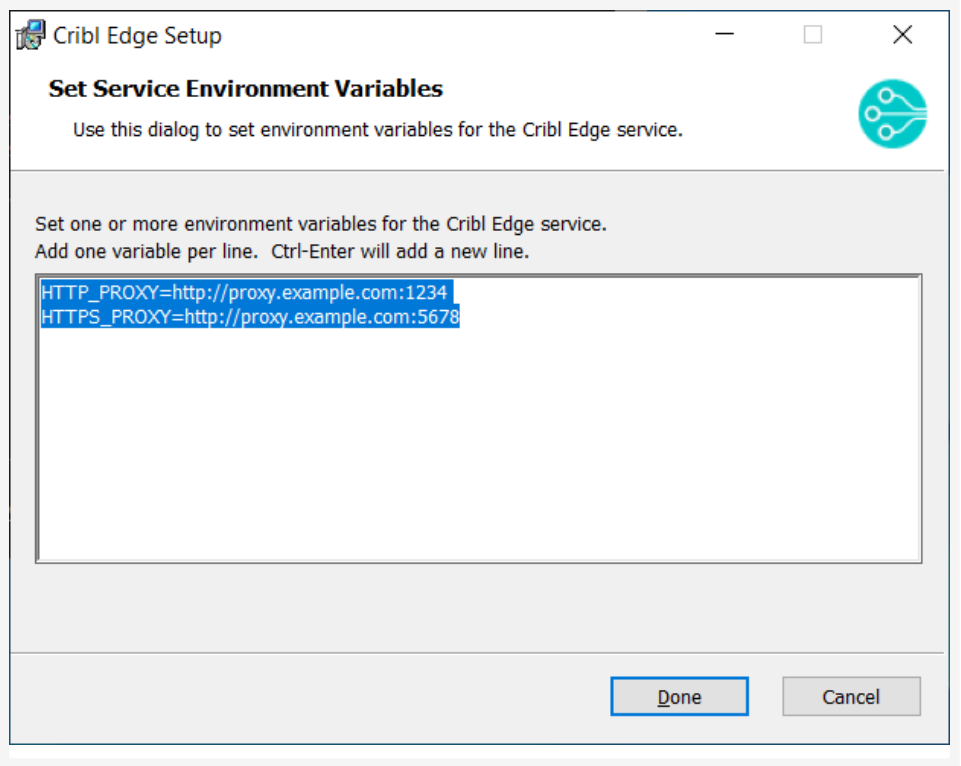


Connection Details

7. Select a Windows account type for the Cribl Edge service:
    - **Local System Account**: The default account type. This is a local Windows service account that does not have a corresponding user profile.
    - **Username / Password**: Enter your **Username** and **Password** credentials to run Cribl Edge. To successfully log Cribl Edge as a service, you might need to include the domain name in the **Username**.



8. Optionally, select **Set Environment Variables** to set environment variables for the Edge service during installation.



Set Environment Variables

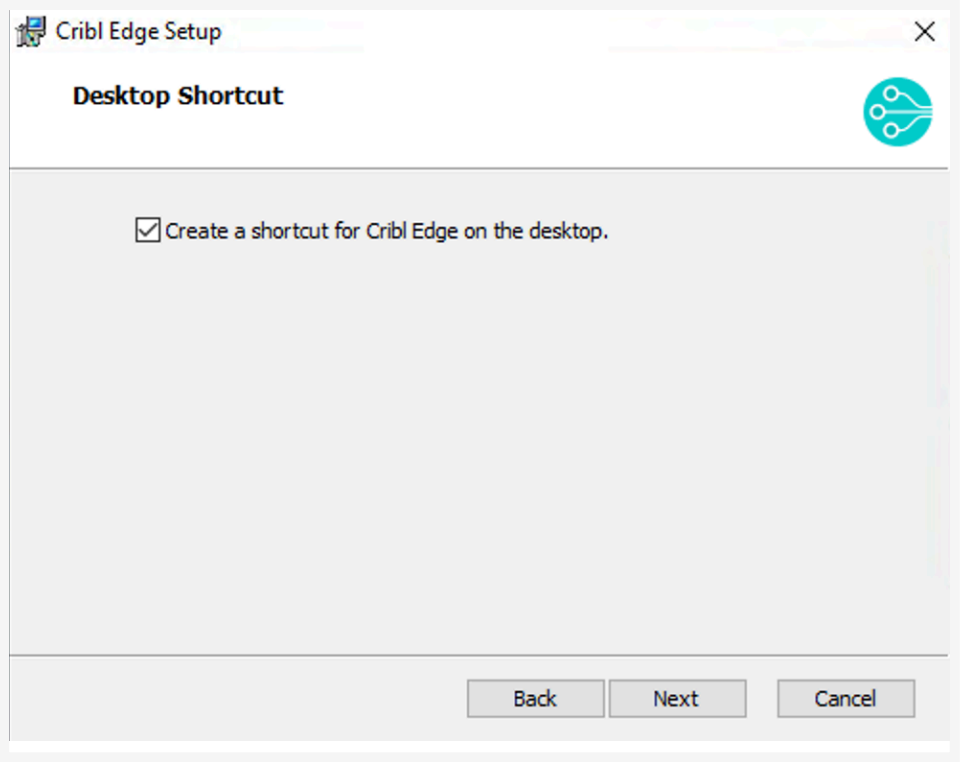
In this example, HTTP and HTTPS proxy settings are being defined to enable Cribl Edge to communicate through corporate proxies, with each variable specified on a separate line.



Setting environment variables is optional and typically not needed for standard installations. This feature is primarily used for special configurations such as configuring proxy settings, enabling FIPS compliance mode, or other advanced scenarios.

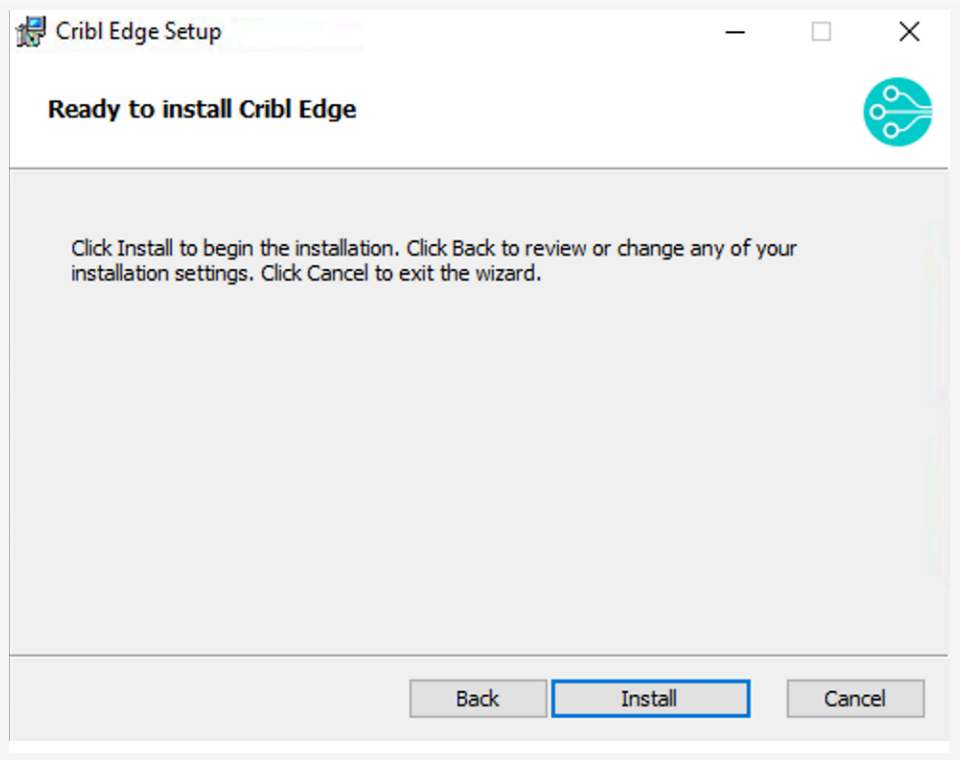
If you've already provided environment variables via the command line, or if there are existing variables, the UI will be pre-populated with these values when you open it. For details, see About Environment Variables above.

9. Optionally, you can create a desktop shortcut.



Desktop shortcut option

10. On the final **Ready to Install** page, click **Install** to confirm.



Installation summary

11. When installation is complete, the Edge Node should report to the Leader.
    In a Single-instance deployment, you can double-click the Cribl Edge icon on your Desktop or File Explorer. This will launch the Cribl Edge login page in your default browser. Or go to `http://localhost:9420` and log in with default credentials (`admin:admin`).

Retrieve Cribl.Cloud Credentials

On Cribl.Cloud, retrieve the **Leader URI** and **Auth Token** from your Cloud instance like this:

1. Navigate to the Manage Edge Nodes page.
2. From the **Add/Update Edge Node** control at upper right, select **Bootstrap new**.
3. Select **Add Windows**.
4. Copy the **Leader or Outpost hostname/IP** value.
5. The **Leader Port** field is `4200`.
6. Display the **Auth Token** value, and copy/paste it into the installer. The **Auth Token** is required to enable communication between the Leader and Edge Node.

Use the MSI Installer

From the command line, you can install Cribl Edge as a single instance or as a Managed Node. See Windows User Permissions and Requirements for important details regarding user accounts and installer settings.



If you want to configure different installation or data directory locations than the default, see Configure Installation Directories.

Install Single-Instance/Standalone Cribl Edge

To run Edge locally, as a single-instance deployment on your own machine, enter the following at your command prompt for a silent install:

```
msiexec /qn /i cribl-<version>-<build>-<arch>.msi
```

Copy

Next, go to `http://localhost:9420` and log in with default credentials (`admin:admin`).

You can now start configuring Cribl Edge with Sources and Destinations, or start creating Routes and Pipelines.

Install Managed Edge Nodes

You can use the Leader UI to concatenate and copy/paste the bootstrap script for adding a Windows Node.

Or, to run Cribl Edge as a managed Node, enter the following at your command prompt for a silent install:

```
msiexec /qn /i cribl-<version>-<build>.msi MODE=mode-managed-edge HOSTNAME=<yourhostr
```

Copy

The installer by default configures the Edge Node to use TLS. If you want to disable TLS, add `TLS=false` to the command.

To set one or more environment variables during installation, use the `SETENVVARS` parameter with pipe-delimited key-value pairs. The entire string must be quoted:

```
msiexec /qn /i cribl-<version>-<build>.msi setenvvars="HTTP_PROXY=http://203.0.113.1:
```

Copy

The variable names and values cannot contain the pipe character ("|") as it's used as the separator.

For prior versions, use the flag `COPYDATA=1` to persist data between installs. `COPYDATA` copies the `instance.yml` file from `C:\Program Files\Cribl` to `C:\ProgramData\Cribl`.

You can now manage this node from the specified Leader.

For other parameter options, see the CLI Reference. Here is an example command:

```
msiexec /qn /i cribl-<version>-<build>.msi  MODE=mode-managed-edge HOSTNAME=192.0.2.1
```

Copy

MSI Installer Properties

Below is a comprehensive list of installer properties specific to the Cribl Edge MSI package, along with their descriptions. These properties allow you to customize your Cribl Edge installation when using the `msiexec` command.

Installer Property	Description
`MODE`	Deployment mode for Cribl Edge. For example, use `mode-managed-edge` for a managed node. If not specified, a Single-instance deployment is implied.
`HOSTNAME`	Leader's hostname or IP address used when installing Cribl Edge in `mode-managed-edge`.

Installer Property	Description
`PORT`	Leader port used when installing in `mode-managed-edge`. The default port is `4200`.
`AUTH`	The authentication token required for the Edge Node to connect to the Leader.
`FLEET`	Fleet to assign the new Edge Node to.
`SETENVVARS`	Multiple environment variables to be defined for the installed Cribl Edge service. Pass a string of pipe-delimited `KEY=VALUE` pairs. This property injects environment variables into the service's configuration at installation time.
`KEEPDATA`	How data is treated when Cribl Edge is uninstalled, then reinstalled. When set to `1`, the installer preserves configuration, state, and logs, ensuring your data is not overwritten.
`USERNAME`	Windows account under which the Cribl Edge service will run. Examples include `LocalSystem` (the default) or a specific user account.
`PASSWORD`	Password for the username, only valid when using a specific user account (not LocalSystem).
`APPLICATIONROOTDIRECTORY`	Installation directory for the Cribl Edge application binaries. This is where the core program files will be placed.
`VOLUMEDIR`	Separate directory for Cribl Edge's runtime data, including configurations, logs, and state information. This allows you to store data independently from the application binaries.
`TAGS`	List of comma-delimited tags to assign to the Edge Node.
`TLS`	TLS for Leader communication. Enabled by default. Set to `false` to disable.

Configure Installation Directories

When installing Cribl Edge on Windows via the MSI installer, you can configure the directories for the application binaries and runtime data. These options are intended for first-time installs only. Changing the installation or data directory is explicitly disallowed during an upgrade. To change directories for an existing installation, you must perform a full uninstall and then start a new installation with the desired paths.

Change the Installation Directory

If you want to change the installation directory during a first-time install, use the `APPLICATIONROOTDIRECTORY` property in your `msiexec` command. For example:

```
msiexec.exe /i cribl-<version>-<build>.msi APPLICATIONROOTDIRECTORY="C:\test\" /log '
```

Copy

If you're using PowerShell to run the silent install, add two extra pairs of quotes to escape the command's own quote delimiters. For example:

```
msiexec.exe /i cribl-<version>-<build>.msi APPLICATIONROOTDIRECTORY=""D:\Custom\Crib
```

Copy

Change the Data Directory Location

By default, Cribl Edge stores its runtime data in `C:\ProgramData\Cribl`. To choose a non-default data directory during a first-time install, use the `VOLUMEDIR` property in your `msiexec` command.

`VOLUMEDIR` specifies the runtime data location, not where application binaries are stored. If the directory doesn't already exist, the installer creates it.

```
msiexec /qn /i cribl-<version>-<build>.msi VOLUMEDIR=D:\CriblData MODE=mode-managed-e
```

Copy

This stores runtime data in `D:\CriblData` instead of the default `C:\ProgramData\Cribl`, while keeping application binaries in `C:\Program Files\Cribl`.

If you completely uninstall and then reinstall Cribl Edge but want to reuse the existing data directory, you can add `KEEPDATA=1` to preserve configuration, state, and logs in that directory. For example:

```
msiexec /qn /i cribl-<version>-<build>.msi  VOLUMEDIR=D:\CriblData MODE=mode-managed-
```

Copy

Set Logging Options

To debug `.msi` installation or upgrade issues, you can run `msiexec.exe` to set logging options. See Microsoft's Logging Options topic.

Change Configuration After Installation

After the installation is complete, if you need to update any settings you configured during installation, make use of the `mode-managed-edge` command (or `mode-edge` for Single-instance deployment).

Running this command resets all the related options. This means that even if you want to change only one option, you also need to provide values for all the other arguments.

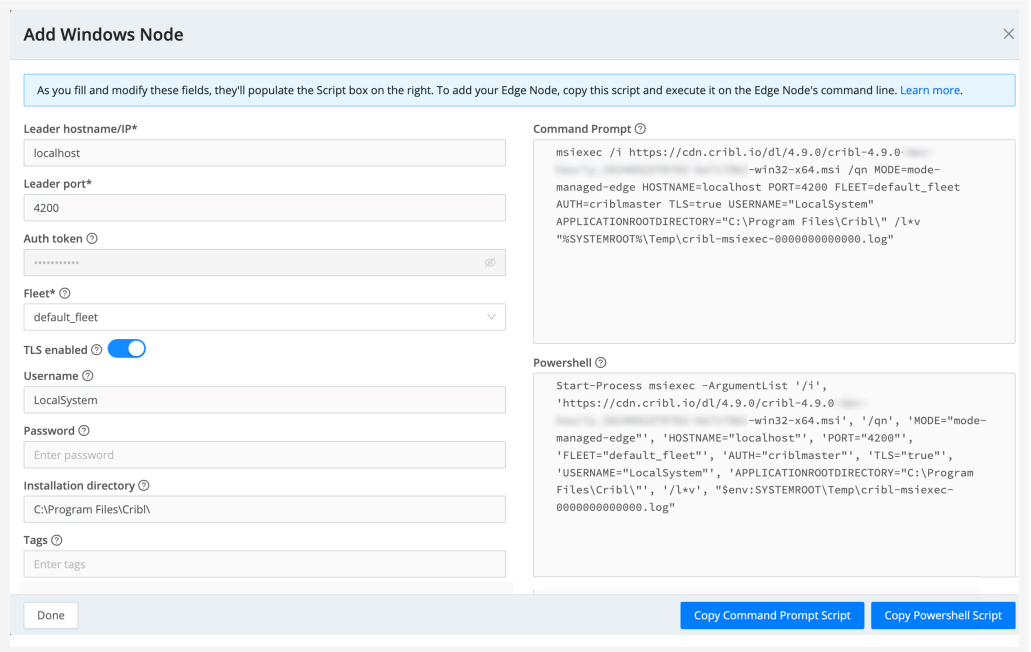
Add or Update a Windows Node

Using the Cribl Edge UI, you can add or update Windows Edge Nodes in your existing deployment.

For more information about adding or updating Edge Nodes, see Adding and Updating Edge Nodes.

To add or update a Windows Edge Node:

1. In your Cribl Edge instance, select **Edge Nodes** from the sidebar (you can also select **Fleets**).
2. Select **Add/Update Edge Node** then **Windows** > **Add** or **Update**.
3. Fill out the fields in the Add Windows Node window. The scripts update in real-time as you change the fields.



Add or update a Windows Edge Node

4. Copy and paste the resulting bootstrap script into a command prompt.
5. The result is a new Edge Node configured with the options you chose. You should see the Edge Node appear in the **Edge Nodes** list.

Log the Time of Installation

The Windows bootstrap script includes a log file with a string of zeros (`/log "%SYSTEMROOT%\Temp\cribl-msiexec-0000000000000.log"`). You can replace the zeros with a timestamp in Unix time (in milliseconds), to log when you first installed the Windows Node.

For example, if you're adding a Windows Node on October 4, 2024 at 6:07:44 PM Pacific Time, update the log file in the script to `/log "%SYSTEMROOT%\Temp\cribl-msiexec-1728090464000.log"`.

This log is included in the Cribl Edge diag, which is helpful for troubleshooting issues with Windows Node deployments.



When the Leader initiates an upgrade of Cribl Edge, the log file name will automatically include Unix time in milliseconds as the timestamp.

Troubleshooting

Here are some common challenges you can encounter when deploying Cribl Edge on Windows.

Upgrading a Pre-4.3.0 MSI Installation

MSIs older than 4.3.0 install Cribl Edge so it belongs to the user running the installer. If you later try to upgrade it as another user, the upgrade can result in a mix of the old and new package contents on disk.

We recommend adding `ALLUSERS=1` to the `msiexec` command when installing or upgrading versions older than 4.3.0 for testing. Example command:

```
msiexec /i cribl-<version>-<build>.msi /qn MODE=mode-managed-edge HOSTNAME=
<yourhostname> PORT=4200 AUTH=myAuthToken FLEET=myfleet USERNAME=LocalSystem
ALLUSERS=1
```

Debugging MSI Installations

For troubleshooting an MSI installation that does not pull configuration settings from the Leader node, you need to know the log path on Windows. The full path depends on where you chose to install.

`$INSTALL_PATH\cribl\log` and `local\edge\` are the log location and conf location, respectively.

Installation Issues with Leader Auth Token

If you encounter problems installing newer versions of Cribl Edge, this may be due to Leader `Auth token` issues. To resolve:

1. Verify the `Auth token` is correct and current in your Leader Node. (You can find the Auth token setting by navigating to **Settings** > **Global** > **System** > **Distributed Settings** > **Leader Settings** > **Auth token**.)
2. Check network connectivity between Edge Nodes and the Leader.
3. Validate successful connections with both Cribl Edge and Cribl Stream products after installation.

Installing or Upgrading Nodes with Endpoint Security

If you are having trouble installing or upgrading Cribl Edge, check for antivirus or other endpoint security tools that may be blocking the `cribl` process (like Windows Defender). To install Cribl Edge, you may need to add temporary or permanent exceptions to these tools.

4.2. Windows User Permissions And Requirements

Learn what user accounts and administrator privileges you need to run Cribl Edge on Windows.

When installing Cribl Edge on Windows, it's important to distinguish between two user accounts:

- **Installer User:** This is the user account you use to run the Cribl Edge installer (MSI). This user must have `administrator` privileges to install the software and create the Cribl Edge service.
- **Service Account:** This is the user account the Cribl Edge service runs under. This account needs appropriate permissions to access your data and APIs, as well as `administrator` privileges to allow the Leader to perform remote Edge Node upgrades across a Fleet. `LocalSystem` is the default account.

Required Administrator Privileges

Ensure you launch the installer (MSI) from a command prompt that is running with full administrator privileges. Simply being logged in as an `administrator` might not be sufficient; you need to explicitly `Run as administrator`.

Service Account Configuration

You have three options to choose from when selecting the service account to run Cribl Edge as: two passwordless options (`LocalSystem` and MSA/gMSA), and local user.

- **LocalSystem**: Cribl Edge on Windows, by default, runs under the `LocalSystem` account (also referred to as `SYSTEM`). This is a predefined system account with extensive privileges on the local machine. The `LocalSystem` account has the required `Log on as a service` privilege by default. It is a passwordless account, meaning it cannot be logged into interactively, and no password is associated with it.
- **Managed Service Accounts (MSA/gMSA)**: You can use Managed Service Accounts (MSA) or Group Managed Service Accounts (gMSA) for the Cribl Edge service. These accounts provide automatic password management and improve security posture. MSA/gMSA are passwordless options. Refer to Microsoft documentation for details: Group Managed Service Accounts Overview.
- **Virtual Service Account**: You can use Cribl Edge with a Virtual Service Account (VSA), a type of Windows-managed local account that allows the service to run with the necessary permissions without requiring a password. Refer to Microsoft documentation for details: Virtual Accounts.
- **Local user**: You can configure Cribl Edge to run under a local user. If you need to run Leader-initiated Edge Node upgrades in a Fleet, add this user to the `Administrators` group. In such cases, the user must also be granted the `Log on as a service` privilege.

Avoid storing passwords in scripts or config files. Exposed credentials jeopardize the local system and any networked resources accessible by that account.

Service Account Username and Password

During installation, you need to specify the username (and in some cases, the password) for the service account. You can do it, for example, by using the `USERNAME` and `PASSWORD` parameters with the `msiexec` command.

One exception is the default service account, `LocalSystem`, which does **not** require a username or password to be specified.

The following is an example `msiexec` command using the `LocalSystem` account (note the absence of `USERNAME` and `PASSWORD` parameters):

```
msiexec /i cribl-<version>-<build>.msi /qn ^
  MODE=mode-managed-edge ^
  HOSTNAME=<leaderHostname> ^
  FLEET=<fleetName> ^
  AUTH=<authToken>
```

Copy

Username with gMSA or VSA

Using gMSA or VSA requires specifying the username in the full form:

- **gMSA**: `domain\gmsaAccountName$`
- **VSA**: `NT SERVICE\Cribl`

These two options do not require you to provide a password.

To install Edge and run the service under a gMSA, you can use the following command:

```
msiexec /i cribl-<version>-<build>.msi /qn ^
  MODE=mode-managed-edge ^
  HOSTNAME=<leaderHostname> ^
  FLEET=<fleetName> ^
  AUTH=<authToken> ^
  USERNAME=<domain\gmsaAccountName$>
```

Copy

To install Edge and run the service under a VSA, use the following command:

```
msiexec /i cribl-<version>-<build>.msi /qn ^
  MODE=mode-managed-edge ^
  HOSTNAME=<leaderHostname> ^
  FLEET=<fleetName> ^
  AUTH=<authToken> ^
  USERNAME="NT SERVICE\Cribl"
```

Copy

Username with a Local User

When running as a local user, selecting **OK** in the installer's user prompt will show the resolved user name. Alternatively, you can find the exact username in the following way:

1. Open the **Services** console (`services.msc`).
2. Locate the Cribl service.
3. Open the service properties and check the **Log On** tab, which shows the name of the user the service runs as.

The following example shows how to specify the local user name and password:

```
msiexec /i cribl-<version>-<build>.msi /qn ^
  MODE=mode-managed-edge ^
  HOSTNAME=<leaderHostname> ^
  FLEET=<fleetName> ^
  AUTH=<authToken> ^
  USERNAME=<domain\user> ^
  PASSWORD=<password>
```

Copy

User Requirements

To successfully run the Edge service under a specific user account, Managed Service Accounts (MSA), Group Managed Service Accounts (gMSA), or Virtual Service Account (VSA), the following requirements must be met.

Group Memberships

The user must be a member of the following groups:

- Event Log Readers
- Network Configuration Operators
- Performance Log Users
- Performance Monitor Users
- Users (default)

User Rights

The user must have the following rights, which can be assigned via Local Security Policy or Group Policy tools:

- Log on as a service
- Manage Auditing and Security Log

Service Accounts and Edge Node Upgrades

When you upgrade Windows Edge Nodes in a Fleet (by setting a target software version), the upgrade is performed by running the MSI installer in the background. If the service account that runs the upgrade does not have administrator permissions, the installer will fail with an "Access Denied" error.

When you run the Cribl Edge service as `LocalSystem` (default), you can run upgrades in a Fleet without additional steps, because `LocalSystem` already has the required permissions to perform upgrades.

To run a Fleet upgrade with MSA/gMSA, add the account to the local administrators group. Configuring MSA/gMSA as an administrator lets you run Edge Node upgrades, but also grants broader privileges to the service account. If you want to avoid it, keep the MSA/gMSA as a non-admin service account and perform Edge upgrades manually. See Upgrade Edge Nodes Manually for information on how to do it.

Troubleshooting

If you encounter issues, check the Windows Event Viewer for error messages related to the installation or the Cribl Edge service. For detailed installer logs, use the `/log logfile.txt` option with `msiexec`. Also, check the Cribl logs themselves for more application-specific information.

4.3. System Proxy Configuration On Windows

Configure the proxy server to handle HTTP/S requests on Windows.

You can direct all outbound HTTP/S requests to go through proxy servers.

Enable Proxy Server

First, you need to make sure the proxy server is enabled.

There are a two ways to do it, using Windows system settings and Group Policy Objects.

Windows Proxy Settings

To enable the proxy server for all processes and services on a host, you can use the system proxy settings:

1. In Windows settings go to **Network & Internet** > **Proxy**.
2. Under **Manual proxy setup**, toggle **Use a proxy server** to **On**.
3. In the **Address** and **Port** boxes, enter the proxy server name or IP address and (optionally) port.
4. Confirm with **Save**.

Group Policy Object

Alternatively, you can use a Windows Group Policy Object to enable the proxy server across groups of accounts or groups of servers. See Microsoft Q&A thread for a detailed instruction.

Communication with Leader over Proxy

If you outbound TCP connection from an Edge Node to the Leader must go through proxy, you need to set the `CRIBL_DIST_WORKER_PROXY` environment variable:

```
CRIBL_DIST_WORKER_PROXY=<socks4|socks5>://<username>:<password>@<host>:<port>
```

HTTP and/or HTTPS?

Several Cribl Edge endpoints rely on the HTTPS protocol - the Cribl telemetry endpoint - which must be accessed with some license types, as well as the CDN used to propagate application updates, certain documentation features (API Reference and docs PDFs), and config bundle downloads.

You might configure certain other Cribl Edge features that require access to HTTP endpoints. For maximum flexibility, consider setting environment variables to handle both the HTTPS and HTTP protocols.

Authenticating on Proxies

You can use HTTP Basic authentication on HTTP or HTTPS proxies. To do this, during installation set the `HTTP_PROXY` and `HTTPS_PROXY` environment variables and specify the user name and password in the proxy URL:

Variable	Value
`HTTP_PROXY`	`http://<username>:<password>@<proxyhost>:<port>`
`HTTPS_PROXY`	`https://<username>:<password>@<proxyhost>:<port>`

Bypassing Proxies with `NO_PROXY`

If you've set the above environment variables, you can negate them for specified (or all) hosts. Add the `NO_PROXY` environment variable to identify URLs that should bypass the proxy server, to instead be sent as direct requests. Use the following format:

```
NO_PROXY="<list of hosts/domains>"
```

Copy

Cribl recommends including the Leader Node's host name in the `NO_PROXY` list.

`NO_PROXY` Usage

Within the `NO_PROXY` list, separate the host/domain names with commas or spaces.

Optionally, you can follow each host/domain entry with a port. If not specified, the protocol's default port is assumed.

To match subdomains, you must either list them all in full (for example, `NO_PROXY=foo.example.com,bar.example.com`), or apply a wildcard by prefixing the domain name with a period or `*.`: `NO_PROXY=".example.com` or `NO_PROXY="*.example.com`.

To match the whole domain including its subdomains, add it both with and without wildcard to the list: `NO_PROXY="example.com,.example.com`.

To disable all proxies, use the `*` wildcard: `NO_PROXY="*"`. `NO_PROXY` with an empty list disables no proxies.

Cloud `NO_PROXY` Usage

You must include any cloud metadata endpoints (such as the AWS Instance Metadata Service) in the `NO_PROXY` list:

- AWS EC2 and Azure VM instances must include `169.254.169.254` in the list. If using IPv6 on AWS EC2, add `fd00:ec2::254` to the list.
- AWS ECS Fargate tasks must include `169.254.170.2`.
- GCP (Google Cloud Platform) VM instances must include `metadata.google.internal` and `169.254.169.254`.

Where Proxies Apply

Proxy configuration is relevant to the following Cribl Edge components that make outbound HTTP/S requests:

Destinations

- S3 Compatible Stores
- Amazon Kinesis Data Streams
- AWS CloudWatch Logs
- AWS SQS
- Azure Blob Storage
- Azure Monitor Logs
- Cribl HTTP
- CrowdStrike Falcon LogScale
- Elasticsearch
- Grafana Cloud
- Honeycomb
- Loki
- Prometheus
- Splunk HEC
- Webhook

Notification Targets

- PagerDuty
- Webhook

Testing Proxies

To initially test your proxy configuration, consider setting up a simple, free proxy server like mitmproxy (https://mitmproxy.org), and then monitoring traffic through that server. Verify that you can trace proxied requests from your Cribl Edge instance, and can validate that outgoing requests (to Destinations) are working properly.

4.4. Windows Troubleshooting

When running Cribl Edge on Windows, you can encounter the following issues. Read on to find the proposed solutions.



For potential issues you can encounter during installation of Windows Edge Nodes, see Windows Installation Troubleshooting.

Windows Event Fields Configuration

If you're encountering issues when collecting Windows events in both `JSON` and `XML` modes, it's likely due to improper configurations. To resolve:

1. Configure Windows events via `XML` format in Cribl Edge (recommended over `JSON`).
2. Build a basic Pipeline/transformation to ensure all expected fields are present.
3. Verify fields are in the desired format after processing.
4. Consider implementing data reduction in your pipeline to optimize usage.

PowerShell Performance Issues with Windows Sources

PowerShell-based collection for System State, Windows Metrics, and Windows Event Log Sources is a legacy option that will be removed in a future release. We highly recommend keeping this option disabled and using the newer native capabilities.

To disable PowerShell-based collection:

1. Upgrade to Cribl Edge `4.13.x` or newer.
2. Disable **Advanced Settings** > **Use Windows Tools**.

The optimized non-PowerShell collection methods provide better performance and reliable data ingestion.

CPU/Memory Issues on Windows

If you notice high CPU/Memory usage on your Windows Cribl Edge Nodes:

- Create a Fleet dedicated to Windows Nodes, and constrain Windows Sources to that Fleet.
- If you want to collect logs only, disable the Windows Metrics Source.

- The Users and Groups collector in the System State Source can trigger high CPU utilization in the Windows `lsass.exe` process where there are many user accounts on the host. This is not uncommon on Domain Controllers. Consider disabling the User and Groups collector if you see that process consuming excessive CPU.

To optimize the Windows Event Logs Source:

- During the Windows Event Log capture, run the collection for 10 seconds and tweak the settings if necessary so that collection doesn't always have to catch up. Consider changing the Windows Events Log Batch Size and Frequency.
  Example command to capture a certain number of events:
  ```
  measure-command {Get-WinEvent -Oldest -MaxEvents 500 @{LogName='Security'} | ForEach-Object -Process {ConvertTo-Json -Compress $_}}
  ```
- For Domain Controllers with heavy log volumes, consider using the Windows Event Forwarder and sending the logs to Cribl Stream, instead of using the Windows Event Log Source in Cribl Edge.
- If you're collecting Windows Events Logs in JSON format, you may experience disk space issues. You can either set XML as the event format on the Windows Events Logs Source, or upgrade Cribl Edge to v.4.8.x or newer and toggle off **Use Windows Tools**. See the Windows Event Logs Source topic for more details.

4.5. FIPS Mode For Cribl Edge (Windows)

This guide will walk you through the process of setting up and running Cribl Edge in FIPS (Federal Information Processing Standards) mode on Windows systems.

FIPS 140-2 or FIPS 140-3 compliance is required for software deployed in U.S. Federal Government and regulated industries. Cribl Edge supports operation in FIPS mode to meet these requirements.

Prerequisites

- Cribl Edge Windows installer (.msi).
- Administrative access to the Windows system.
- FIPS-validated OpenSSL provider (instructions below).

What You Need to Know About Pre-Configuration



Before configuring FIPS-enabled Edge Nodes, ensure your Leader environment is already configured and running in FIPS mode.

- When running in FIPS mode, Cribl Edge uses only FIPS-approved cryptographic algorithms, ensuring all data handling meets Federal security standards.
- When connecting to a FIPS-configured Leader, Cribl Edge must also be in FIPS mode for full compliance.

Set up Cribl Edge in FIPS Mode

To set up Cribl Edge in FIPS mode, you'll need to build your own OpenSSL FIPS provider. For details, see OpenSSL: Notes for Windows platforms.

Build Your Own FIPS Provider

This section guides you through the process of building your own OpenSSL FIPS provider.

Install Required Tools

1. Install Strawberry Perl, which includes the Netwide Assembler – (NASM).
2. Microsoft C++ toolchain (Visual Studio).
3. If you have `git` installed, you may need to rename/move its `perl.exe` to avoid conflicts.

Build the Provider

1. Find and open the Visual Studio (20XX) Native Tools Command Prompt for `x64` architecture. You can search for it, by typing `x64` in the **Start** menu. Right-click on it and select **Run as administrator**. (Do not use PowerShell or plain CMD).
2. Download the latest FIPS-validated version of OpenSSL:

   ```
   curl -LO https://www.openssl.org/source/openssl-3.0.9.tar.gz
   tar -xf openssl-3.0.9.tar.gz
   cd openssl-3.0.9
   ```

   Copy

   If you run into an error message: `ERROR: module machine type 'x86' conflicts with target machine type 'x64'`, it means you're not using the right version of the command prompt for what you're trying to do. Make sure you've selected the `x64 Native Tools Command Prompt`.
3. Run the following commands in the Visual Studio Developer Command Prompt (with `x64` architecture selected):

   ```
   perl configure VC-WIN64A enable-fips
   nmake
   ```

   Copy

   The build process might take considerable time to complete. Once the build is complete, you'll have a `providers` directory. The providers directory is created within the OpenSSL build directory (for example, `openssl-3.0.9/providers/`).
4. Once built, you'll need the following files from the build directory:
   - `providers\fipsmodule.cnf` - This specific file from your build is essential as it contains a unique hash that validates the DLL you built.
   - `providers\fips.dll`
   - `providers\legacy.dll`

Set up the Provider

This step creates a dedicated directory for Cribl's FIPS-related files and copies the necessary DLL and configuration files into it, preparing the application for FIPS-compliant operation.

In the same Native Tools Command Prompt, type the following:

```
mkdir "C:\Program Files\Cribl-FIPS"
copy providers\fips.dll "C:\Program Files\Cribl-FIPS"
copy providers\legacy.dll "C:\Program Files\Cribl-FIPS"
copy providers\fipsmodule.cnf "C:\Program Files\Cribl-FIPS"
```

Copy

Generate the `nodejs.cnf` Configuration File

This step creates a necessary configuration file that enables FIPS compliance for the `Node.js` environment used by Cribl. You have two options.

Option 1: Manual Configuration Creation

1. Open an Administrative Command Prompt
2. Run the following command to create and open the configuration file:

```
notepad "C:\Program Files\Cribl-FIPS\nodejs.cnf"
```

Copy

3. Copy and paste the following configuration content into the file:

```
nodejs_conf = nodejs_init
openssl_conf = nodejs_init
.include "C:\\Program Files\\Cribl-FIPS\\fipsmodule.cnf"

[nodejs_init]
providers = provider_sect
alg_section = algorithm_sect

[provider_sect]
default = default_sect
fips = fips_sect

[default_sect]
activate = 1

[algorithm_sect]
default_properties = fips=yes
```

Copy

4. Save the file and close Notepad.

This configuration redirects `Node.js` to use FIPS-compliant cryptographic modules and algorithms. The file includes references to the FIPS module configuration and sets default properties to enforce FIPS compliance across all cryptographic operations.

Option 2: Using the Cribl CLI (Post-Installation)

Alternatively, after installing Cribl Edge, you can generate the configuration file using the Cribl CLI:

```
cribl generateFipsConf -d <fully qualified path to fipsmodule.cnf>
```

You can generate the `nodejs.cnf` configuration once on any machine and then distribute it to others, eliminating the need to repeat this step on each individual machine.

Install Edge with FIPS Configuration

To install Cribl Edge with FIPS mode enabled:

1. Use the `SETENVVARS` parameter with the MSI installer:

```
msiexec /i CriblEdge-<version>.msi SETENVVARS="OPENSSL_MODULES=C:\Program Files\Cribl
```

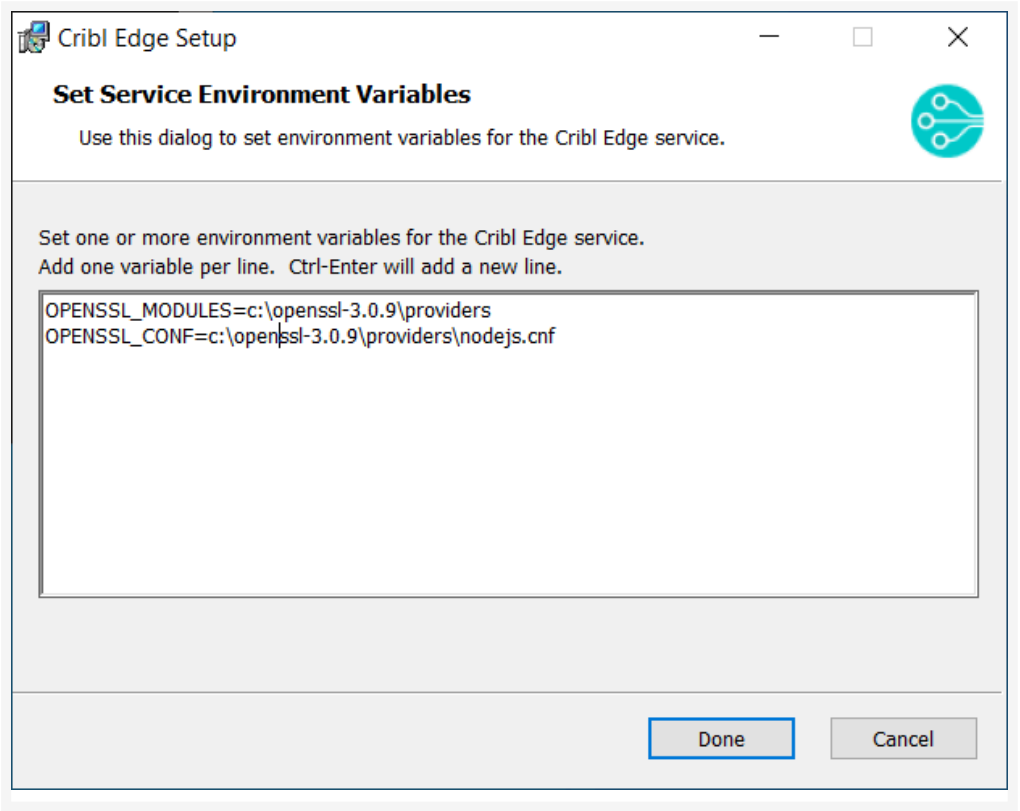
Copy

2. This opens the Windows Installer Wizard. Choose `Managed Mode` and complete the configuration to connect to the FIPS-enabled Leader.

When executing the MSI installer command, pay attention to these important parameter requirements:

- The pipe character (|) separates multiple environment variables.
- Make sure to enclose the entire `SETENVVARS` string in quotes.
- `OPENSSL_MODULES` points to the directory containing the provider files.
- `OPENSSL_CONF` points to the specific configuration file.

Alternatively, you can set the environment variables through the Installation Wizard:



Set Environment Variables

For details, see Installing Cribl Edge on Windows.

Confirm Cribl Edge is in FIPS Mode

After installation, validate that Edge is running in FIPS mode by checking for FIPS initialization messages:

1. On the Leader or Edge Node, search `cribl.log` for this message:

```
"level":"info","message":"running with FIPS enabled", "OpenSSLVersion":"3.0.13+quic
```

Copy

2. In the UI, navigate to: **Node Settings** > **Diagnostics** > **System Info**.

The presence of the above message confirms that Cribl Edge is in FIPS mode.

Troubleshooting

5. Cribl Edge On MacOS

Learn about running Cribl Edge on macOS.

Cribl Edge on macOS offers easy-to-use tools for exploring and collecting macOS events. You can run Cribl Edge on a supported macOS machine to collect logs and track local files.

Edge on macOS does not include support for system metrics or system state at this time. See below for a more detailed list of limitations.

Limitations

Cribl Edge on macOS is subject to the following limitations.

Modes

Cribl Edge on macOS supports only the following modes:

- `Edge: Single`
- `Edge: Managed Edge (managed by Leader)`

This means you can't switch Cribl Edge on macOS into Cribl Stream mode (Single-instance, Worker, or Leader).

You can, however, switch between the Cribl Edge supported modes via the UI, in **Settings** > **Distributed Settings** > **Mode**.



Do not select an unsupported mode from this drop-down! Doing so will cause the Cribl service to fail.

Sources and Destinations

Cribl Edge on macOS supports the same Sources and Destinations as Cribl Edge on Linux, with the following exceptions.

Unavailable Sources

The following Sources are unavailable on macOS

- System Metrics Source
- System State Source

Unavailable Destinations

The following Destination is unavailable on macOS:

- Amazon Security Lake Destination

Other Limitations on Sources and Destinations

Other limitations include:

- The Kafka Destination does not support Kerberos authentication on macOS.
- The File Monitor Source only allows **Manual** discovery mode, not **Auto**.

Functions

The Grok Function is unavailable on macOS. If you include it in a Pipeline, Cribl Edge processing will skip over it.

Data Formats

Cribl Edge on macOS does not support reading or writing Parquet files. This is a limitation on the following Destinations: Amazon S3, Azure Blob Storage, MinIO, and FileSystem/NFS.

Goat-to Resources

One agent, no headache: Cribl Edge now supports macOS Blog

5.1. Install Cribl Edge On MacOS

Install and deploy Cribl Edge Nodes on-prem on macOS.

You can install Cribl Edge in a supported macOS environment to collect logs and track local files.

Supported MacOS Versions

Cribl Edge supports the following macOS versions:

- 26.x (Tahoe)
- 15.x (Sequoia)
- 14.x (Sonoma).

You can install Cribl Edge using the same universal binary on both Intel and Apple silicon processors.

Before You Install Cribl Edge on MacOS

1. Set a compatible browser as your default browser: Microsoft Edge, Firefox, Safari, or Chrome. (We support the five most-recent versions of these browsers. Cribl Edge is not compatible with Internet Explorer.)
2. Ensure that the required ports are available (see Ports).
3. Go to the Cribl Download page and set the **Software** drop-down to `Cribl Edge for macOS`.
4. Select **Download now** to get the `.pkg` installer.

You can also install a macOS Edge Node in a Distributed deployment by running the bootstrap script in your command prompt. Run the bootstrap script with `sudo`.

Install Cribl Edge

From the command line, you can install and configure Cribl Edge to act either as a Single-instance deployment, or as a managed Node.

1. To install Cribl Edge from the package you downloaded, enter the following at your command prompt:

```
sudo installer -pkg <package-file>.pkg -target /
```

Copy

This will install Cribl Edge to `/opt/cribl/`.



To prevent issues with Cribl file operations, `/opt/cribl` and all its subdirectories must reside on the same device. Mounting separate devices within `/opt/cribl` is not recommended. For external storage needs, such as lookups or persistent queues (PQs), create a completely separate directory outside of `/opt/cribl`.

2. Set `/opt/cribl` as your `$CRIBL_HOME` directory.



To make the `$CRIBL_HOME` environment variable available on your command line, you can:

- Assign it once, using the `export` command: `export CRIBL_HOME=/opt/cribl`
- Set it as a default, by adding it to your to your terminal profile file.

3. Set the installation mode to `mode-managed-edge` to connect the Node to the designated Leader:

```
sudo /opt/cribl/bin/cribl mode-managed-edge /
   -H <leader-hostname-or-IP> /
   -p <port> /
   -u <token> /
  [-g <fleet>]
```

Copy

- `<token>` must match the Leader token, to ensure a connection with the Leader. Get the Leader token from **Settings** > **Global** > **System** > **Distributed Settings** > **Leader Settings** > **Auth token**.
- The new Node will be assigned to a Fleet on the Leader, either `default_fleet`, or a different one according to Fleet mappings.
- You can manually point to a specific Fleet by providing its name as `-g <fleet>`, but Fleet mappings will override this choice.



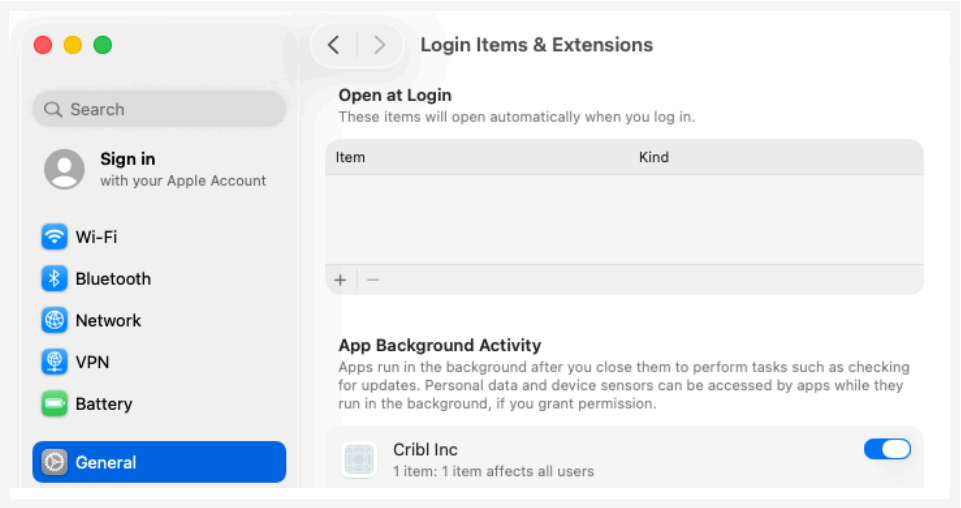
If you only need a Single-instance deployment of Cribl Edge (without a Leader), do not change the mode. Cribl Edge will run in `mode-edge` automatically.

4. Restart the Cribl service:

```
sudo /opt/cribl/bin/cribl restart
```

Copy

5. Allow Cribl to run in the background. In macOS system settings, select **General**, then **Login Items & Extensions**. Make sure that Cribl is enabled in **App Background Activity**.



Enabling Cribl for background activity in macOS settings

6. You can now log in to your Leader and locate the newly installed Cribl Edge Node and start configuring Sources and Destinations, or creating Routes and Pipelines.

If you installed a single instance of Cribl Edge without a Leader, go to `http://localhost:9420` (or another port you defined) and log in with default credentials (`admin:admin`).

For more ways to configure Leader and Edge Nodes, see Set Up Leader and Edge Nodes.

5.2. Deploy Cribl Edge On MacOS With JAMF

Deploy Cribl Edge on macOS machines managed with JAMF Pro.

If you are using JAMF Pro to manage macOS machines, you may choose to deploy Cribl Edge using JAMF as well. You can do it by getting a bootstrap script for adding Edge Nodes from your Leader, and then configuring a JAMF policy that will run it.

See Add and Update Edge Nodes for more information about the bootstrap script.



Before you start, ensure that your environment meets the OS and System Requirements.

Create Fleet for Edge Nodes on MacOS

First, create a new Fleet that will house your macOS Edge Nodes.



Cribl generally recommends creating separate Fleets for different OS platforms.

1. In the sidebar, select **Fleets**, then select **New Fleet**.
2. Give the Fleet a descriptive **Name**, for example, `MacLaptops`, and confirm with **Save**.

Prepare Bootstrap Script

Now, prepare the bootstrap script that will be used to add new Edge Nodes.

1. In the sidebar, select **Fleets**, then select your new `MacLaptops` Fleet.
2. Select **Add/Update Edge Node**, then **macOS**, then **Add**.
3. Check that the target Fleet, the Leader hostname or IP, port, auth token, and any other options have been prefilled correctly, and correct them if necessary.
4. Copy the generated script.

Use the Bootstrap Script in JAMF

Now you will run the script through JAMF Pro to create new Edge Nodes.

1. Log in to your JAMF Pro console.

2. Navigate to **Settings** in the sidebar. Then, in the **Computer management** section, select **Scripts**.
3. Select **New**, give the new script a name (for example, “Install Cribl Edge”) and add an optional description.
4. On the **Script** tab, enter the bootstrap script you copied from Cribl Edge.
5. Confirm with **Save**.

Configure JAMF Pro Policy

Finally, you need to ensure JAMF runs the script on the desired macOS machines.

1. In the JAMF Pro console, navigate to **Computers** > **Policies**.
2. Select **New** to create a new policy and give it a descriptive name.
3. In the **Trigger** section, enable **Recurring Check-in** to ensure the policy triggers with the recurring check-in frequency.
4. In **Execution Frequency**, select **Once per computer** to ensure JAMF does not attempt to reinstall Cribl Edge.
5. Select **Scripts**, then in the list of available scripts locate the “Install Cribl Edge” script and select **Add**.

With this configuration, JAMF will run the script that deploys Cribl Edge once for every machine it manages, adding the new Edge Nodes to the `MacLaptops` Fleet.

6. Cribl Edge In Containers

You can deploy Cribl Edge in containerized environments using Docker or Kubernetes.

Running Cribl Edge in a container ensures consistent deployment, simplifying installation and upgrades. Containerization enables easy scaling and management via orchestration tools like Kubernetes.

Read the following topics for more information about running Cribl Edge in a container:

Topic	Description
Run Cribl Edge in a Docker Container	Install Cribl Edge in a container using a Docker image.
Deploy Cribl Edge via Kubernetes	Deploy Cribl Edge in Kubernetes Pods to collect logs and metrics.
Upgrading Containers	Upgrade a Cribl Edge instance in a container.

6.1. Run Cribl Edge In A Docker Container

Install Cribl Edge in a container using a Docker image.

You can easily deploy and run Cribl Edge from a container image into a Docker container, making it simple to integrate Cribl Edge into your existing containerized environments.

In this article, you'll learn how to pull the Cribl Edge Docker image, configure it, and manage its lifecycle within your containerized environment.

Add a Docker Node from the Cribl Edge UI

As a best practice, we recommend using the UI to create and copy/paste the bootstrap script for adding a Docker node. See our documentation on how to Add a New Edge Node with a Bootstrap Script.

Start the Container with a Docker Command

For a Single-instance deployment, start the container with the Docker command below (replace `4.13.2` in the last line with the version of Cribl Edge you want to install):

```
docker run -d
 -e CRIBL_EDGE=1 \
 -p 9420:9420 \
 -v /var/run/docker.sock:/var/run/docker.sock \
 -v /:/hostfs:ro \
 --privileged \
 --restart unless-stopped \
 --name cribl-edge \
 cribl/cribl:4.13.2
```

Copy

For a Distributed deployment, edit the Docker command below to replace `INSERT_TOKEN` with a unique, secure token value. Then start the container with the revised command.

```
docker run -d \
 -e CRIBL_DIST_MODE=managed-edge \
 -e CRIBL_DIST_LEADER_URL=tcp://INSERT_TOKEN@leaderhere:4200?group=fleet_name \
 -e CRIBL_EDGE=1 \
 -p 9420:9420 \
 -v /var/run/docker.sock:/var/run/docker.sock \
 -v /:/hostfs:ro \
 --privileged \
 --restart unless-stopped \
 --name cribl-edge \
 cribl/cribl:4.13.2
```

Copy

With a Leader on Cribl.Cloud, encryption is enabled by default. Set the hybrid worker's `CRIBL_DIST_LEADER_URL` environment variable to begin with the `tls://` protocol. For example:

```
CRIBL_DIST_LEADER_URL=tls://<token>@logstream-<tenant>.cribl.cloud:4200
```

Copy

Once that's running, the UI displays on `http://localhost:9420/`

You can now start configuring Cribl Edge with Sources and Destinations, or start creating Routes and Pipelines.

When a Node in a Kubernetes Fleet restarts, it might receive a new GUID, which can cause duplicate entries to appear on the **Edge Nodes** page. To resolve this issue, navigate to **General Settings** > **Fleet Configuration** and set **Time to keep disconnected Nodes** to 0.

Find Edge Nodes Running in a Container

Edge Nodes running in a container report additional metadata about the container to the Leader. This helps you quickly identify which of your Edge Nodes are running in a container from the Leader UI and map them to the appropriate Fleets.

When Cribl Edge is running in a container, you will see the `hostOs` property in events collected by the System State Source. This object will contain properties about the container when they're available, such as:

- `arch`
- `cpu_count`

- `cpu_speed_mhz`
- `cpu_type`
- `interfaces`
- `memory`
- `platform`
- `release`
- `hostname`
- `timezone`

From the **Mappings** menu, you will also see the `hostOs` property in the **Edge Node info** panel. This object will include the `id`, `addresses`, and `version` of the container when available.



`interfaces` is a set of key-value pairs that contain an array of CIDR values.

Run Cribl Edge as a Non-Root User

You can run the Cribl image as a non-root user within a container using Docker's `--user` option or the user setting in Compose. You can specify any desired `UID` (User ID).

To ensure correct functionality, you must set the `CRIBL_VOLUME_DIR` environment variable to a location the non-root user can write to. This is because the software requires write permissions to directories like `/opt/cribl/state`, `/opt/cribl/log`, and `/opt/cribl/local`. Failure to set `CRIBL_VOLUME_DIR` will result in a warning and potential functional issues.

If you mount a persistent volume, setting `CRIBL_VOLUME_DIR` to the mount point is strongly recommended for optimal performance and reliability. While `/tmp/cribl_data` might be used, a dedicated persistent volume ensures data integrity and prevents potential data loss or corruption.

We recommend specifying `CRIBL_VOLUME_DIR` whenever running as a non-root user to avoid unexpected behavior.

See example below:

```
docker run -d \
 -u 999:999
 -e CRIBL_DIST_MODE=managed-edge \
 -e CRIBL_DIST_LEADER_URL=tcp://INSERT_TOKEN@leaderhere:4200?group=fleet_name \
 -e CRIBL_EDGE=1 \
 -p 9420:9420 \
 -e CRIBL_VOLUME_DIR=/tmp/ \
 -v /var/run/docker.sock:/var/run/docker.sock \
 -v /:/hostfs:ro \
 --restart unless-stopped \
 --name cribl-edge \
 cribl/cribl:4.13.2
```

Copy



To prevent issues with Cribl file operations, `/opt/cribl` and all its subdirectories must reside on the same device. Mounting separate devices within `/opt/cribl` is not recommended. For external storage needs, such as lookups or Persistent Queues (PQs), create a completely separate directory outside of `/opt/cribl`. While `/opt/cribl` is the default installation path, Cribl can be installed to other locations if necessary.

Override Default Ports

You can override default ports in the following configuration files:

- Edge UI port (`9420`): Default definitions for `host`, `port`, and other settings are set in `$CRIBL_HOME/default/cribl/cribl.yml`, and can be overridden by defining alternatives in `$CRIBL_HOME/local/cribl/cribl.yml`.
- Data Ports, for example HTTP In (`10080`), TCPJSON in (`10420`) have default definitions for `host`, `port` and other settings defined in `$CRIBL_HOME/default/cribl/inputs.yml`, and can be overridden by defining alternatives in `$CRIBL_HOME/local/cribl/inputs.yml`.



In the case of an API port conflict, the process will retry binding for 10 minutes before exiting.

Update the Docker Image and the Packaged OS

For details, see the following topics:

- Updating the Docker Image

- Updating the Packaged OS

Shut Down Docker Containers

To shut down Docker containers that are running Cribl Edge, use `docker stop` instead of `docker kill`. This allows Docker to shut down the container cleanly. When you run `docker stop` for your Cribl Edge containers, the following happens:

- The container runs `cribl stop` and waits for all `cribl` processes to exit before shutting down.
- Docker waits up to 10 seconds by default. If this is not enough time for your processes to exit, you can increase the default time using `docker stop -t 30 <container>`. Replace the number `30` in this example with the amount of time you need, in seconds.

Troubleshooting

- When you deploy containers via Kubernetes in `managed-edge` mode, this Node will - after applying the

6.2. Deploy Cribl Edge Via Kubernetes

Deploy Cribl Edge in Kubernetes Pods to collect logs and metrics.



Cribl Edge supports both Docker and `containerd` runtimes.

Considerations

Cribl Edge connects to the Kubernetes Cluster API and the `kubelet` API on each node to read logs and metrics. To prevent under- or over-collection of logs and metrics, you must deploy the Cribl Edge agent as a DaemonSet in your Kubernetes cluster. Ensure that Cribl Edge is able to detect that it's running inside a Kubernetes Pod, by setting the `CRIBL_K8S_POD` environment variable.

To run the Kubernetes Sources (Metrics, Logs) inside a cluster, use the default `ServiceAccount`, and mount the connection details in `var/run/secrets/kubernetes.io/serviceaccount`. If you want the Source to look in a different directory, use the `CRIBL_SERVICEACCOUNT_PATH` environment variable to set the new path to the `ServiceAccount`.

Optionally, you can run the Kubernetes Sources (Metrics, Logs) outside of a cluster, by specifying connection details in a `kubeconfig` file (like `kubectl`). Either place the file in `$HOME/.kube/config`, or use the `$KUBECONFIG` variable to point to it.

If you are enabling role-based access control (RBAC) in your Kubernetes cluster, Cribl Edge requires a `ClusterRole` to authorize the Service Account to read logs and metrics. This is a `read-only` role, which does not require `write` permissions. Plan ahead with your Kubernetes admins and security team to authorize the installation of this `ClusterRole`.

If your Kubernetes deployment doesn't use valid certificates, then set the `CRIBL_K8S_TLS_REJECT_UNAUTHORIZED` environment variable to `0` to disable that expectation. When you disable this environment variable, all Kubernetes features (including Metadata, Metrics, Logs, and workload instrumentation metadata) will tolerate invalid TLS certificates (i.e., expired, self-signed, and so on.) when connecting to the Kubernetes APIs. If this environment variable is not defined or set to `1` and the certificate validation fails, then all connection attempts to the Kubernetes API will be rejected. For details, see Which Certificate Does kubelet Use?

All other Cribl Edge Kubernetes components can run inside a namespace of your choice. This page uses the example namespace `cribl-edge`.

Configure a Service Account

You can configure the Leader deployment to use a Kubernetes ServiceAccount for accessing cloud resources. This approach proves especially beneficial when you need to grant IAM roles directly to the Leader pod. A common use case is leveraging AWS KMS as the KMS provider without the need for access key IDs and secrets.

To configure a ServiceAccount for the Leader chart, you can modify the following options within your `values.yaml` file or by using the `--set` parameters during Helm deployment:

Value	Default	Description
`serviceAccount.create`	`false`	If set to `true`, the chart will create a new ServiceAccount specifically for the Leader Pod.
`serviceAccount.name`	`undefined`	Name of the ServiceAccount to use: • If `create: true` and this value is unset, the name defaults to `<release>-leader`. • If `create: true` and this value is set, the service account will use the provided name. • If `create: false`, this value **must** match the name of an existing ServiceAccount.
`serviceAccount.annotations`	`{}`	A key/value map of annotations to be added to the newly created ServiceAccount.

When `serviceAccount.create` is set to `false` and a `name` is provided, the chart assumes you are referencing an existing, pre-configured ServiceAccount. Conversely, if `serviceAccount.create` is `true`, the chart will generate a new ServiceAccount, either with a default name or the name you specify. When creating a service account, you can include annotations to declare the specific IAM roles that the service account should have permission to access.

Example configuration for creating a ServiceAccount with IAM role annotations:

```
serviceAccount:
  create: true
  name: "cribl-leader-sa"
  annotations:
    [eks.amazonaws.com/role-arn](https://eks.amazonaws.com/role-arn): "arn:aws:iam::1
```

Copy

Deploying

To deploy the Cribl Edge agent in your Kubernetes cluster, you can use our Helm charts. These charts provide the minimum configuration required to install Cribl Edge correctly.

Our deployment has been tested across multiple Kubernetes platforms on several cloud service providers. The Cribl Helm Charts provide a fast, repeatable, and consistent way of deploying and upgrading an entire Kubernetes environment.

You can customize our Helm charts with additional configurations required for your Kubernetes platform - such as annotations for load balancers, environment variables for secrets, etc. For details, follow the link above.

A simple deployment example using Cribl's Helm chart:

```
helm repo add cribl https://criblio.github.io/helm-charts/

helm install -n cribl-edge --set "cribl.leader=tcp://<token>@leader:4200" cribl-edge
```

Copy

You can also use the UI to concatenate and copy/paste the bootstrap script for adding a Kubernetes node.

If you are building your own manifests, the DaemonSet pods must be configured with the following environment variables:

- `CRIBL_DIST_LEADER_URL: <changeme>`
- `CRIBL_DIST_MODE: managed-edge`
- `CRIBL_EDGE: true`
- `CRIBL_K8S_POD :<must match the name of the pod>`

Tolerations

By default, Kubernetes will not schedule a Pod on a node that has taints. You must configure your deployment to allow scheduling on these nodes; otherwise, Cribl Edge will be unable to collect logs and metrics from them.

The defaults in our Helm chart's `values.yaml` file are configured like this:

```
tolerations:
  - operator: Exists
```

Copy

Kubernetes CPU Limit Configuration

The Cribl Edge Helm chart ensures that the number of Worker Processes is determined by the CPU resources allocated to the pod, rather than the physical CPU count on the host.

To achieve this, it sets the `CRIBL_K8S_CPU_LIMIT` environment variable to the value of the `resources.limits.cpu` setting on the pod. Cribl Edge then uses the value as a CPU limit to determine how many Worker Processes to start.

We recommend maintaining the default configuration of the `CRIBL_K8S_CPU_LIMIT` environment variable, to prevent performance problems like resource contention and pod instability from exceeding CPU limits.

Updating Helm Charts for OpenShift or Non-Root Environments

To make sure your deployments run smoothly on OpenShift or non-root environments, you'll need to configure the security context through the `values.yaml` file. The `values.yaml` file allows you to override default settings and customize your deployment without altering the core Helm chart templates.

To add or update the `values.yaml` :

1. Locate or create a local `values.yaml` file.
2. Modify the `securityContext` section.
3. Customize the capabilities to match your environment's needs:

```
securityContext:
  capabilities:
    add:
      - CAP_NET_BIND_SERVICE
      - CAP_SYS_PTRACE
      - CAP_DAC_READ_SEARCH
      - CAP_NET_ADMIN
```

Copy

4. Install your updated values to Helm, using this command: `helm install -f /bar/values.yaml`

For a description of the capabilities, see Set Capabilities for Cribl Edge.

RBAC

Cribl Edge's Kubernetes Logs and Kubernetes Metrics Sources use the Kubernetes API to read logs and metrics from the cluster. If you have role-based access control (RBAC) implemented, you will need to authorize the Cribl Edge Pod Service Account to read the relevant data.

Kubernetes Events Source

To collect cluster-level events, the Kubernetes Events Source needs watch access to the Kubernetes API.

An example RBAC rule is:

```
- apiGroups:
    - "events.k8s.io"
  resources:
    - events
  verbs: ['watch']
```

Copy

Kubernetes Logs Source

The Kubernetes Logs Source requires the ability to get, list, and watch Pods in all namespaces.

An example RBAC rule is:

```
- apiGroups:
    - ""
  resources:
    - pods
  verbs: ['get', 'list', 'watch']
```

Copy

Kubernetes Metrics Source

The Kubernetes Metrics Source reads metric information from many resources in your cluster. The minimum permissions required are:

```
- apiGroups:
    - ""
  resources:
    - configmaps
    - endpoints
    - limitranges
    - namespaces
    - nodes
    - persistentvolumeclaims
    - pods
    - replicationcontrollers
    - secrets
    - services
    - strategicMergePatches
    - nodes/log
    - nodes/metrics
    - nodes/proxy
    - nodes/spec
    - nodes/stats
  verbs: ['get', 'list', 'watch']

- apiGroups:
    - "apps"
  resources:
    - daemonsets
    - deployments
    - replicasets
    - statefulsets
  verbs: ['get', 'list', 'watch']
- apiGroups:
    - "batch"
  resources:
    - cronjobs
    - jobs
  verbs: ['get', 'list', 'watch']
- apiGroups:
    - "autoscaling"
  resources:
    - horizontalpodautoscalers
  verbs: ['get', 'list', 'watch']
- apiGroups:
    - "policy"
```

```
  resources:
    - poddisruptionbudgets
  verbs: ['get', 'list', 'watch']
- apiGroups:
    - "networking.k8s.io"
  resources:
    - ingresses
    - networkpolicies
  verbs: ['get', 'list', 'watch']
- apiGroups:
    - "admissionregistration.k8s.io"
  resources:
    - mutatingwebhookconfigurations
    - validatingwebhookconfigurations
  verbs: ['get', 'list', 'watch']
- apiGroups:
    - "certificates.k8s.io"
  resources:
    - certificatesigningrequests
  verbs: ['get', 'list', 'watch']
- apiGroups:
    - "storage.k8s.io"
  resources:
    - storageclasses
    - volumeattachments
  verbs: ['get', 'list', 'watch']
```

Copy

Prometheus Edge Scraper

To discover a Kubernetes Node or Pods, the Prometheus Edge Scraper Source must be able to list Pods across all namespaces.

An example RBAC rule is:

```
- apiGroups:
    - ""
  resources:
    - pods
  verbs: ['list']
```

Copy

Health Checks

An example health-check configuration for Kubernetes is:

```
readinessProbe:
    httpGet:
        path: /api/v1/health
        port: 9420
        scheme: HTTP
    initialDelaySeconds: 15
    timeoutSeconds: 1
livenessProbe:
    httpGet:
        path: /api/v1/health
        port: 9420
        scheme: HTTP
    initialDelaySeconds: 15
    timeoutSeconds: 1
```

Copy

If you have Fleets running Edge Nodes with version 4.8.2 and older, you can encounter failing health checks that require specific API server configuration.

By default, the Cribl Edge API listens on port 9420, instead of on Cribl Stream's default 9000 port. This requires special care when you configure your Edge Fleet in the Cribl UI.

On Fleets running Edge Nodes with version 4.8.2 and older, the API server is configured be default to listen on the loopback interface (`127.0.0.1`). For your Nodes to remain reachable, you must change the listening interface address to `0.0.0.0` in **Fleet Settings** > **General Settings** > **API Server Settings** > **General**. This will bind to all IPs in the container. If you do not make this change, your health checks will start failing after the Edge agent downloads its initial config bundle.

Another side effect of not changing the Listening Interface address, is the Edge Node will not appear on the Leader UI.

Environment Variables

You can use the following environment variables to configure Cribl Edge on Kubernetes:

- `CRIBL_SERVICEACCOUNT_PATH` - by default, set to: `/var/run/secrets/kubernetes.io/serviceaccount`.
- `CRIBL_K8S_POD` - the name of the Kubernetes Pod in which Cribl Edge is deployed.
- `KUBECONFIG` - set this for local testing.
- `CRIBL_K8S_TLS_REJECT_UNAUTHORIZED` - set to `0` to disable certification validation when connecting to the Kubernetes APIs. When you disable this environment variable, all Kubernetes features (including Metadata, Metrics, Logs, and workload instrumentation metadata) will tolerate invalid TLS certificates (i.e., expired, self-signed, etc.) when connecting to the Kubernetes APIs.

See also our list of all Cribl Environment Variables.

Upgrading Kubernetes Edge Nodes

For details, see Upgrading Kubernetes Edge Nodes.

Uninstalling

To remove the Cribl Edge agent from your Kubernetes cluster, run the following command:

```
helm uninstall -n <namespace> <releasename>
```

The `helm uninstall` command automatically deletes all deployed resources associated with the Cribl Edge Helm Chart.

Troubleshooting

Here are some common challenges you can encounter when deploying Cribl Edge via Kubernetes.

Duplicated Nodes in Node List

Edge Nodes in Kubernetes restart with new GUIDs, causing the **Edge Nodes** page to track and display each restart as a separate Node. To prevent this behavior in Fleets containing Kubernetes nodes, set **General Settings** > **Fleet Configuration** > **Time to keep disconnected Nodes** to 0. This ensures that Nodes in these Fleets will not be tracked between restarts.

Certificate Validation Fails

If your Kubernetes deployment doesn't use valid certificates, certificate validation will fail and all connection attempts to the Kubernetes API will be rejected.

To avoid this, set the `CRIBL_K8S_TLS_REJECT_UNAUTHORIZED` environment variable to 0 to disable the expectation of valid certificates. When you disable this environment variable, all Kubernetes features (including Metadata, Metrics, Logs, and workload instrumentation metadata) will tolerate invalid (for example, expired or self-signed) TLS certificates when connecting to the Kubernetes APIs.

Log Loss Due to Polling Interval and Rotation

The Kubernetes Logs Source polls the local kubelet for Pod discovery. While this is a lightweight operation, the

7. Set Up Cribl Edge

Cribl Edge offers different types of deployments that you can choose based on your needs:

- Cribl.Cloud, including a hybrid option.
- Customer-managed deployments, including:
    - A Single-instance deployment, where you deploy a single Edge Node on Linux, Windows, or macOS.
    - A Distributed deployment, where you set up Leader and multiple Edge Nodes.

You can also run Cribl Edge on Docker, or deploy it via Kubernetes.

To prepare for a deployment and gather decision-making information, consult Deployment Planning.

7.1. Get Started With Cribl Edge

Quickly set up a working instance of Cribl Edge.

This quick start guide shows you how to get Cribl Edge running.

You can either start with a free Cribl.Cloud instance, or deploy the Cribl Edge agent on a physical or virtual machine. We'll cover both options below, so feel free to jump to the relevant section in this article:

- Download and Install Cribl Edge.
- Quick Start with a Cloud Instance.

Check the Minimum Deployment Requirements

Before you proceed, make sure that the hosts for all Cribl Edge agents - and for any Leader that you install on your own infrastructure - meet the System Requirements that we list for Linux, Windows, and macOS.

Download and Install Cribl Edge

To install Cribl Edge, select the Linux or Windows tab below and follow the corresponding steps.

Linux Windows macOS



To avoid permissions errors, install and run Edge as the same Linux user. For details on creating a new user, see Enabling Start on Boot.

1. Navigate to the Cribl Downloads page.
2. Select either `x64` or `ARM` Cribl Suite from the drop-down. You can also copy and paste one of the provided `curl` scripts into a command-line interface.
3. Open a command-line interface and navigate to the `/opt/` directory:

```
cd /opt/
```

Copy

4. Un-tar the resulting `.tgz` file in the `/opt/` directory.

```
tar xvzf cribl-<version>-<build>-<arch>.tgz
```

Copy

For example:

```
tar xvzf cribl-4.3.1-12f82b6a-linux-x64.tgz
```

Copy

Edge is now installed in a `cribl` subdirectory, by default: `/opt/cribl/`.

5. Move it to the `cribl-edge` directory with:

```
mv /opt/cribl/ /opt/cribl-edge
```

Copy

6. In your terminal, go to `/opt/cribl-edge/bin` and switch your installation to Edge mode:

```
./cribl mode-edge
```

Copy

Run Cribl Edge

Now you can start, stop, and verify the Cribl Edge server using these `./cribl` CLI commands:

- **Start**: `./cribl start`
- **Stop**: `./cribl stop`
- **Get status**: `./cribl status`

For other available commands, see CLI Reference.

1. In your browser, open `http://127.0.0.1:9420` and log in with default credentials (`admin`, `admin`).
2. Register your copy of Cribl Edge when prompted.
   After registering, you'll be prompted to change the default password.

Quick Start with a Cloud Instance

As mentioned above, the Leader Node can reside in a free Cribl.Cloud account. You'll get a ready-to-use Cribl Leader that includes Cribl Edge functionality. For a detailed comparison, see Cribl.Cloud vs. Self-Hosted.

To get a Cribl.Cloud account:

1. Go to https://cribl.cloud/signup/ and register an account.
2. On the **Create Organization** page, optionally enter an **Organization Name** (a friendly alias for the randomly generated ID that Cribl will assign to your Organization).
3. Select an AWS Region to host your Cribl.Cloud Leader. Cribl currently supports the following Regions:
    - `US West (Oregon)`
    - `US East (Virginia)`
    - `US East (Ohio)`
    - `Europe (Frankfurt)`
    - `Europe (London)`
    - `Europe (Zurich)`
    - `Asia Pacific (Sydney)`
    - `Asia Pacific (Singapore)`
    - `Canada (Central)`

For details on Cribl.Cloud Organizations, refer to Cribl.Cloud documentation:

- Register Cribl.Cloud Organization.
- Manage Cribl.Cloud organization.

Add Edge Nodes

You now have a Cribl Leader running with Cribl Edge capabilities enabled, but still need to add Edge Nodes. To do this:

1. From Cribl Edges's top nav, select **Manage** > **Edge Nodes**.
2. On the resulting **Edge Nodes** tab, click **Add/Update Edge Node** at the upper right and then select **Linux** > **Add**
3. Copy the resulting script to your clipboard and close the modal.

4. Paste the script onto your Edge Node's command line and execute it.



To add an Edge Node on a Windows or using Docker or Kubernetes, refer to Add and Update Edge Nodes.

What's Next?

Now that you have Cribl Edge running, you can try out the following resources:

- Follow a tutorial to set up your first Sources, and learn how to forward data to Destinations.
- Explore Cribl Edge on Linux or Windows to learn more about the various capabilities of Cribl Edge by taking a tour through its UI.

7.2. Cribl.Cloud Vs. Self-Hosted

Learn how a Cribl.Cloud deployment differs from an on-prem installation.

A Cribl.Cloud deployment differs in several ways from a customer-managed (on-prem) deployment of Cribl Suite software on your own infrastructure. Keep in mind these differences as you choose how to launch Cribl Edge. Also consider these differences as you navigate the product's UI, in-app help (including tooltips), and documentation.

Cribl Edge and Cribl.Cloud

A Cribl.Cloud deployment lets you connect your Edge Nodes to a Cloud-hosted Leader Node.

In contrast with Cribl Stream, where you have the option to use Cribl-managed Worker Groups, Cribl Edge Fleets can only be customer-managed. They still benefit from all the features of running a Cloud-hosted Leader Node that are listed on this page.

Cloud-Only Features

Certain Cribl products and features are available **only** on Cribl.Cloud:

- Cribl Search, an application for searching, exploring, and analyzing machine data in place and at API endpoints.
- Cribl Lake, a data lake solution for long-term, full-fidelity data storage.
- Workspaces, an option for isolating parallel Cribl Leaders with separate access controls and other configurations.

For details about how all Cribl products interoperate to manage data, see Cribl Reference Architecture, Full-Suite.

Simplified Administration

Cribl.Cloud has been designed with options to accommodate everyone - from first-time evaluators to Enterprise customers who manage a worldwide network of private-cloud, public-cloud, and/or data-center deployments.

Cribl.Cloud's Free offering is designed to help you launch Cribl Edge - and to start processing data - as quickly and easily as possible. Upgrading to a paid Standard or Enterprise plan provides expanded deployment and configuration options.

For a comparison of features in Free, Standard, and Enterprise Cribl.Cloud plans, see Pricing.

Simplified Distributed Architecture

Cribl.Cloud is preconfigured as a Distributed deployment. A Free or Standard plan allows only a single Fleet, and some Distributed Settings cannot be configured.

Git Preconfigured

Without an Enterprise plan, the **Settings** > **Global** > **System** > **Git Settings** section is omitted. However, a local `git` client is preconfigured in your Cribl.Cloud Organization. On the top nav, use the **Version Control** button (with a branched symbol) to commit/push changes to `git`. Select **Deploy** to deploy your committed changes. Cribl.Cloud does not support Git remote repos.

Automatic Restarts and Upgrades

Without an Enterprise plan, the **Settings** > **Controls** link are omitted. Cribl handles Leader and Fleet restarts automatically on your behalf.

Simplified Access Management and Security

In Cribl.Cloud, you can manage access control for your Organization by selecting **Organization** in the sidebar and then selecting **Members & Teams**. The options on this tab will vary depending on your plan.

If you have a Cribl.Cloud Enterprise plan, you can use the Key Management Service (KMS), which maintains the keys Cribl Edge uses to encrypt secrets on Fleets and Edge Nodes. Go to **Settings** > **Security** > **KMS** to configure KMS.

If you add an Enterprise Plan, cloud and hybrid Leaders support Local and Google SSO authentication, along with OpenID Connect (OIDC) and SAML federated authentication. Cribl.Cloud does not currently support LDAP.

Permission- and Role-based access control (RBAC) is simplified in Cribl.Cloud. For details, see Permissions.

Security Features Comparison: Self-Hosted vs. Cribl.Cloud

The following table outlines the security responsibilities for self-hosted and Cribl.Cloud deployments, highlighting key differences in deployment security, data protection, threat detection, and access control.

Feature	Self-Hosted	Cribl.Cloud
Deployment Security	Customer responsible for securing the deployment environment (network isolation, physical access control, user access management).	Cribl manages the security of the cloud infrastructure.
Configuration Security	Customer responsible for securing configuration files and tokens.	Cribl manages the security of configuration files and tokens.
Git Configuration Security	Customer responsible for securing the Git repository.	Cribl manages the security of the Git repository.
Data at Rest	Customer responsible for encryption and key management.	Encrypted at rest by using industry standard encryption (e.g., AES-256) for storage services.
Data in Motion	Customer configures encryption (TLS) for data in transit.	Preconfigured TLS for some Sources. Can be further configured.
Threat Detection & Response	Customer responsibility. Use Cribl Stream security features (limited) and external tools.	Cribl bolsters its security posture with internal security teams and an external MSSP, ensuring comprehensive protection for its production and corporate environments through threat detection, workload scanning, and vulnerability management.
Patch Management	Customer responsible for applying security patches to Cribl Stream and underlying infrastructure.	Cribl manages patching of Cribl.Cloud (Cribl-managed) infrastructure.
Access Control	Customer configures user Roles and Permissions (RBAC).	Simplified RBAC with Enterprise plans offering advanced options (SSO, KMS).
Authentication	Customer configures authentication methods.	Local and SAML/OIDC IDP with Enterprise plans (additional options).
Key Management	Customer manages encryption keys.	Enterprise plans offer Key Management Service (KMS) for key storage.
Compliance	Customer responsible for adhering to compliance standards.	SOC 2 Type II compliant and GDPR-compliant (Cribl.Cloud).

Transparent Licensing

The Cribl.Cloud sidebar does not display a **Settings** > **Global** > **Licensing** link, nor does the **Monitoring** > **System** submenu include **Licensing**. Your plan is managed by your Cribl.Cloud Organization, and you can check credits and usage history in the FinOps Center.

Other Simplified Settings

These features are available only in on-prem deployments:

- **Settings** > **Global** > **Scripts** (if enabled - Cribl.Cloud does not support configuring or running shell scripts).

Configuring persistent queues in a Cribl.Cloud deployment requires an Enterprise plan. You can freely define the **Queue size limit** on your Edge Nodes, based on the disk space you provision.

Available Ports and TLS Configurations

To get data into Cribl.Cloud, your Cribl.Cloud Organization provides several Sources and ports already enabled for you, plus 11 additional TCP ports (`20000-20010`) that you can use to add and configure more Cribl Edge Sources. Additionally, external port `443` is available for TCP-based services—it maps internally to port `10443`. For details, see Cribl.Cloud Ports.

TLS Details

TLS encryption is pre-enabled for you on several Sources, also indicated on the your Workspace's **Data Sources** tab.

Cribl HTTP and Cribl TCP Sources/Destinations

Use the Cribl HTTP Destination and Source, and/or the Cribl TCP Destination and Source, to relay data between Edge Nodes that are either connected to the same Leader or different Leaders. For more information about transferring data across Organizations, go to Transfer Data Between Workspaces or Environments.

7.3. Learn About Connected Environments

If you have one or more on-prem Cribl Edge environments and Cribl.Cloud Organization(s), you can configure **Connected Environments**.

Connected Environments enables you to:

- **Take advantage of Cribl.Cloud.** You can use Connected Environments to connect an on-prem Leader to Cribl.Cloud, then use Cribl.Cloud infrastructure management capabilities, the credit-based billing model, as well as Cribl Lake and Cribl Search solutions. For example, you have on-prem environments but you want to use the credit billing model, and you want to send Cribl metric data to Cribl Lake/Search. From there, you can build dashboards to monitor Cribl throughput and performance.
- **Send data from an on-prem deployment to a Cribl.Cloud deployment.** If you previously had an on-prem deployment of Cribl and you are moving forward with a Cloud-first or Cloud only solution, you can use connected environments to onboard your data from on-prem Edge Nodes to Cribl.Cloud Edge Nodes and transfer data without incurring double billing costs. See Send Data from On-Prem to Cribl.Cloud for implementation details.



For full access to Connected Environments, upgrade Cribl Edge to 4.8.2 or newer.

Accomplish this by creating a connection between the on-prem Leader and the Cribl.Cloud Leader. Then, use the **Connections** page in your Cribl.Cloud account to manage credit consumption for all of your on-prem environments from one Cribl.Cloud interface. This feature is called Universal Subscription.

This section provides instructions for connecting on-prem Leaders to Cribl.Cloud.

Understand Environment Compatibility

Sender → Receiver	Org/Workspace scenario	Cribl HTTP/TCP Support	Receiver Billed for Ingest?	Example Deployments	Notes
Same Leader	Same Workspace		No	• On-prem Worker Group A → Worker Group B under the same on-prem Leader. • Cribl.Cloud hybrid Worker	This is the default Cribl to Cribl pattern.

Sender → Receiver	Org/Workspace scenario	Cribl HTTP/TCP Support	Receiver Billed for Ingest?	Example Deployments	Notes
				Group A → Cribl.Cloud managed Worker Group B in the same Workspace. • Cribl Edge Fleet → Cribl Stream Worker Group.	
Different Leaders	Same on-prem license key		No	• Two on-prem Stream deployments sharing a license key. • Edge Fleet under Leader 1 → Stream Worker Group under Leader 2, same key.	Treated as a single paid environment via shared key.
Different Leaders	Two Workspaces in the same Cribl.Cloud Org		No	• Cribl.Cloud-managed hybrid Worker Group in Workspace A → Cribl.Cloud-managed Worker Group in Workspace B. • Edge → Stream across Workspaces in the same Org.	Requires supported versions and shared Org – uses auth tokens on 4.15+.
Different Leaders	Different Cribl.Cloud Orgs		Yes	Enterprise customer Org A → MSSP Org B via Cribl HTTP/TCP.	Supported, but the receiving Org is billed. No single-ingest across Orgs.

Sender → Receiver	Org/Workspace scenario	Cribl HTTP/TCP Support	Receiver Billed for Ingest?	Example Deployments	Notes
On-prem → Cribl.Cloud	No Connected Environments or Universal Subscription. Same paid Org/license.		No	On-prem Cribl Stream or Edge → Cribl.Cloud-managed Worker Group for central processing (same license/org).	Standard Edge → Stream or Stream → Stream “hub” architectures.
On-prem → Cribl.Cloud	Different paid Orgs, no Universal Subscription.		Yes	Hybrid Worker Group for Customer A → Managed Worker Group in MSP Org B.	Counts as two separate paid environments.
Hybrid Worker Group → Cribl.Cloud Hybrid or Managed Worker Group	Same Organization		Yes	Cribl.Cloud Hybrid Worker Group in Workspace A → Cribl.Cloud-managed Worker Group in Workspace B.	Typical “hybrid WG in Workspace A → Cloud WG in Workspace B” expansion path.
Hybrid Worker Group → Cribl.Cloud Hybrid or Managed Worker Group	Different Organization		Yes	Hybrid Worker Group for Customer A → Managed Worker Group in MSP Org B.	Counts as two separate paid environments.
On-prem Cribl Edge or Stream → Cribl.Cloud Hybrid Worker Group	Same Org with Universal Subscription		Yes	On-prem Edge Fleet with Universal Subscription → Cribl.Cloud Hybrid Worker Group	Not supported for single-ingest billing. Treated as separate environments.

Sender → Receiver	Org/Workspace scenario	Cribl HTTP/TCP Support	Receiver Billed for Ingest?	Example Deployments	Notes
On-prem Cribl Edge or Stream → Cribl.Cloud-managed Worker Group	Same Org with Universal Subscription		Yes	On-prem Worker Group with Universal Subscription → Cribl.Cloud-managed Worker Group in a Workspace.	US governs billing of on-prem env; Cloud ingest is billed normally unless also covered by shared key/Org rules.
Cross-environment flow on free tier	N/A		Not supported	Free-tier Workspace ↔ paid Org.	Free tier license or subscriptions don't qualify for single-ingest benefit.

Before You Begin Connecting On-Prem Leaders

There are a few tasks you must complete before you can start connecting your on-prem Leader to Cribl.Cloud:

1. Create a Cribl.Cloud account.
   To set up connected environments, you must have an Enterprise Cribl.Cloud account.
2. Decide how you want to connect to the Leader.
   There are two methods you can use to connect an on-prem Leader to a Cribl.Cloud Leader:
   - From the Cribl.Cloud account in **Connections** > **Connected Environments**.
   - From the on-prem Leader in **Distributed Settings** > **Cloud Connection**.
3. Purchase a Cribl.Cloud Enterprise Plan with credits.



Credits serve as the virtual transaction currency in the Cribl product suite. Each Cribl product has a predefined usage rate, and as you use the products, credits are deducted from your initial pool.

Configuration Considerations

Take these considerations into account for your on-prem Leader, allowlists, security groups, and Workspaces:

7.3.1. How To Connect On-Prem Leaders To Cribl.Cloud

To connect existing on-prem Cribl Edge Distributed deployments to Cribl.Cloud, select an on-prem Leader that is running Cribl Edge 4.8.2 or newer, then follow the steps.



If you have multiple environments, such as staging and production environments when using GitOps, you must connect each environment individually.

1. On the top bar, select **Products**, and then select **Cribl**.
2. In the sidebar, select **Workspace**, and then select **Connections**.
3. Select **Configure New Environment** and fill out the following fields:

Field	Description	Example
Environment name	A name for your environment. This name will display on the **Connected Environments** page. You can search for and filter environments by name.	Goat Farm East Production
Workspace hostname/ID	Hostname of your Cribl.Cloud Workspace. This field is read-only.	`example-goat-farm-prod.cribl.cloud`
Workspace port number	Port number that the Workspace Leader uses to communicate with the on-prem Leader.	`4200`
Workspace token	The auth token of the Cribl.Cloud Leader. This field is read-only and is automatically populated.	`a12bcDEFgHiJKLmnOpQRstuvw3x4z56`

Script type	The format of the script to connect to the on-prem Leader:
	◦ Environment variable
		Environment variable:
		`CRIBL_DEPLOYMENT="Goat Farm West Product`
		`CRIBL_CLOUD_WORKSPACE_URL="tls://a12bcDE`
		CLI:

Field	Description	Example

	◦ CLI	`./cribl cloud-workspace`
		`-d “Goat Farm West Production”`
		`-H example-goat-farm-prod.cribl.cloud`
		`-p 4200`
		`-u a12bcDEFgHiJKLmnOpQRstuvw3x4z56`

Use a Command-Line Script

If you’re using the CLI option, follow these steps, in order:

1. Select **Copy Script**.
2. Open a command-line window in the Leader environment that you want to connect to Cribl.Cloud.
3. Navigate to the `$CRIBL_HOME/bin` subdirectory.
4. Paste the script results onto the command line, and run the script.

Use an Environment Variable

The **Environment Variable** script populates with the details you enter when you’re configuring a new environment. For example:

```
CRIBL_DEPLOYMENT="On-prem Stream"
CRIBL_CLOUD_WORKSPACE_URL="tls://a12bcDEFGHIjKLMnopPqr123@cribl-cloud-account-0a1bcde
```

Copy

If you’re using the environment variable, copy its output, and set it when - or before - you run `cribl start`. How you set this variable depends on your deployment.

- For example, if your Cribl Edge deployment is running in a container, set the environment variable in the container via run call or configuration.
- If you’re running your own server, you can set environment variables for the `cribl` user, or you can declare the variable along with the `start` command. For details, see our CLI Reference.

High Availability Leader Configuration

Before you configure a connected environment with a High Availability (HA) Leader, ensure you are comfortable setting up and managing Leader HA. See [Leader High Availability/Failover](deploy-add-second-leader for more information.

When you set up the primary Leader to connect to Cribl.Cloud in an HA environment, set the `CRIBL_DEPLOYMENT` and `CRIBL_CLOUD_WORKSPACE_URL` environment variables. You can do this in the Cribl.Cloud UI or using CLI.

For example:

```
Environment=CRIBL_DEPLOYMENT=aws_leader_prod
Environment=CRIBL_CLOUD_WORKSPACE_URL=tls://large-goat-farm.cribl.cloud:4200
```

Copy

The failover Leader will inherit the configuration from the filesystem changes made by the primary Leader.

Enable the On-Prem Leader Connection to Cribl.Cloud

You can connect a Leader Node to an existing Enterprise Cribl.Cloud Organization.

1. In the sidebar, select **Settings**.
2. On the **Global** tab, select **Distributed Settings**.
3. Select **Cloud Connection**.
4. Toggle **Join Cribl.Cloud Workspace** on.
5. Fill in the required fields:

Field	Description
Connection URL	The **Cribl.Cloud URL** to connect to this Leader. 1. In your Cribl.Cloud Organization, locate the URL field value on the sidebar **Workspace** > **Access** tab. 2. Copy the URL and paste it into the **Connection URL** field, using the format: `main-random-goat-a12bcdef.cribl.cloud`.
Connection port	The port for the Cribl.Cloud Leader. Default is `4200`.
Connection auth token	The authentication token for the Cribl.Cloud Leader. You'll need to get this from the Cribl.Cloud environment: 1. Navigate to **Workspace** > **Connections** and select **Configure New Environment**. 2. Copy the value in the **Workspace token** field.

Field	Description
	3. In the on-prem environment, paste the Workspace token from Cribl.Cloud.

6. Select **Save**.
7. In the Cribl Subscription Services Agreement modal, select **Agree** to accept the agreement.
8. In the confirmation modal, select **Yes** to save changes and restart.



Once you connect an on-prem Leader to a Cribl.Cloud Organization, the license type changes to a Cribl.Cloud billing account. This means the **Licensing** page will not be accessible. To return to using an on-prem license, toggle **Join Cribl.Cloud Workspace** off, select **Save**, and select **Yes** in the confirmation modal to save changes and restart.

7.3.2. Simplify Billing With Universal Subscription

This feature requires an Enterprise Plan.

If you have multiple instances of Cribl Edge on-prem, you're likely managing multiple licenses. Instead of letting license management run your life, you can take advantage of the Universal Subscription feature. Universal Subscription lets you manage an existing on-prem license with Cribl.Cloud credits.

To get started, connect your on-prem Leader to your Cribl.Cloud Leader to send billing data payloads. Once your environments are connected, you can use the FinOps Center to get a unified, all-product view of your Cribl Edge usage and spending.

This replaces an on-prem license, and gives you the flexibility to maintain your on-prem deployments without managing multiple licenses and tracking spend across environments.

An on-prem Cribl Edge environment consists of at least one Leader Node (the control plane) and one or more Fleets.

Connecting your on-prem Leader license to Cribl.Cloud billing allows you to:

- Monitor credit consumption for each of your on-prem Cribl Edge environments from one user interface.
- Create a billing- and usage-based connection between the Cribl Edge on-prem Leader and the Cribl.Cloud Leader, which receives metrics from the on-prem Leader. For information about the data payloads, visit Data Payloads for Connected Environments.

Configure Connected Environments

For the setup procedure, go to How to Connect On-Prem Leaders to Cribl.Cloud, then come back here to explore how to view credit and usage data for your on-prem Connected Environments.

Full access to Connected Environments requires you to upgrade Cribl Edge to 4.8.2 or newer.

Explore Credit Consumption for Connected Environments

Once you set up a Connected Environment, you can view the incoming usage data and credit consumption for your on-prem and Cloud deployments of Cribl Edge. You must be an Organization Owner to view this data.

For more information on Cribl.Cloud billing, go to the FinOps Center page.

View Usage Data

1. In the sidebar, select **Organization**, then **FinOps Center**.
2. Select **Usage Overview**.
3. Go to **Per Product Consumption** and expand either Cribl Edge or Cribl Stream.
4. Find **Connected < Product > GBs In**.

View Credit Consumption

1. In the sidebar, select **Organization**, then **FinOps Center**.
2. On the **FinOps Center** submenu, select **Invoices**.
3. Select a month to expand the cost items. Select **Stream** or **Edge**.
4. View **Connected < Product > GBs In** usage and cost.

Here, you will see each on-prem environment that is connected to this Cribl.Cloud Organization, and the credits each environment is consuming.

Limitations of the Universal Subscription Feature

Connected Environments has some usage limitations:

- The Connected Environments feature is available only on Enterprise Cribl.Cloud plans. For Enterprise plan details, see Pricing.
- Single-instance deployments of Cribl Edge aren't eligible for a Universal Subscription.
- The Distributed deployment on-prem Leader must be running Cribl Edge 4.8.2 or newer.

7.3.3. Data Payloads For Connected Environments

In a Connected Environment setup, on-premises Leaders connect to a Cribl.Cloud Leader in the same way that Edge Nodes connect to a Leader. The Cribl.Cloud Leader sends down a license, and the on-premises Leader sends up a heartbeat that includes telemetry, which encompasses metrics and other metadata.

Why Is the Telemetry Important?

The telemetry sent from an on-premises Leader to a Cribl.Cloud Leader is encapsulated in an event. This event serves as a heartbeat, confirming that the connected environment is active and functioning properly. It includes key metadata such as the system's status, timestamp, configuration details, and metrics. Cribl.Cloud uses this data to:

- Monitor connectivity between the on-premises Leader and the cloud.
- Track system health by evaluating fields like status, edgeNodes, and metrics.
- Ensure compliance with licensing by verifying the config.featuresRev field.
- Provide insights into data flow, such as input and output metrics.

How Often Is the Heartbeat Sent?

The heartbeat interval is determined by the **Task heartbeat period** field, which specifies the number of seconds between each event. In the provided example below, the heartbeat is sent every `10` seconds. This interval is configurable, meaning administrators can adjust it to optimize performance based on their needs. For details, see Task Manifest and Buffering Limits in Fleet Settings.

What Happens if the Heartbeat Is Missed?

If a heartbeat event is not received within the expected interval, the following actions may occur:

- **Retries** - The on-premises Leader may attempt to resend the heartbeat after a short delay.
- **Alerting** - Cribl.Cloud can generate an alert if the heartbeat is missing for an extended period, indicating a potential issue.
- **Failure detection** - A prolonged absence of heartbeats may be interpreted as a disconnection or system failure, prompting further investigation or automated recovery steps.

On-Premises Leader to Cribl.Cloud Leader Payload

This section describes the structure of the event payload sent from an on-premises Leader to a Cribl.Cloud Leader in a Connected Environment.

Sample Event

Here is a sample event that Cribl.Cloud received from a connected environment.

```
{
  "__criblEventType": "event",
  "type": "info",
  "_time": 1737577024.567,
  "status": "healthy",
  "localTime": 1737577024,
  "config.hbPeriodSeconds": 10,
  "config.featuresRev": "YbK7TIGe4jp2rdujnFiJqq0OBV67r2FKGvrHr4C6I5g=",
  "edgeNodes": 50,
  "__socketAddr": 192.168.1.100:443,
  "__srcIpPort": 10.0.0.5:52314,
  "metrics": [
    [
      0,
      0,
      "total.in_bytes",
      233,
      1737578280038,
      1737578340038,
      [
        "ci",
        "tcp:in_tcp",
        "input",
        "tcp:in_tcp",
        "input_type",
        "tcp",
        "__worker_group",
        "default",
        "__dist_mode",
        "worker",
        "__worker_node",
        "46731aec-d220-4cb0-9971-0dd0a7907d54",
        "__product",
        "stream"
      ],
      null,
      "float32"
    ]
  ],
  "__raw": "{\"type\":\"info\",\"_time\":173757...}"
}
```

Copy

Event Payload Fields

The following table describes the important fields in the event payload.

Field	Usage	Example
__criblEventType	Type of event	event (always)
type	Type of event	info (always)
_time	Timestamp of the event in Unix time (seconds)	1737577024.567
status	Status of client	healthy
localTime	Local timestamp in Unix time (seconds)	1737577024

7.3.4. Send Data From On-Prem To Cribl.Cloud

You can use Connected Environments to send data from an on-prem deployment to a Cribl.Cloud deployment. For example, if you previously had an on-prem deployment of Cribl and you are moving forward with a Cloud-first or Cloud only solution, you can use connected environments to onboard your data from a on-prem Edge Nodes to their counterpart Cribl.Cloud Edge Nodes.

You can transfer data from on-prem Edge Nodes to the Cribl.Cloud Edge Nodes without incurring any data transfer costs.

Overview of the Data Migration Workflow

The workflow for migrating data from on-prem Edge Nodes to Cribl.Cloud Edge Nodes has these phases:

1. Create a Connected Environment. This process generates the information you need to set up the data flow between the on-prem Edge Nodes and the Cribl.Cloud Edge Nodes.
2. On-Prem Edge Nodes: Configure HTTP or TCP Destination. Add an auth token on the on-prem Edge Nodes to configure the first half of the connection between the on-prem Destination and the Cribl.Cloud Source.
3. Cribl.Cloud Edge Nodes: Configure HTTP or TCP Source. Add a matching auth token on the Cribl.Cloud Edge Nodes to complete the connection.
4. Validate the Data Flow. Confirm that the data is properly flowing from the on-prem Edge Nodes to the Cribl.Cloud Edge Nodes.

Create a Connected Environment

To begin the process of migrating data from on-prem to Cribl.Cloud, you need to first set up a Connected Environment on the on-Prem Leader. This process generates the information you need to set up the data flow between on-prem Edge Nodes and the Cribl.Cloud Edge Nodes.

To set up a Connected Environment, follow the instructions on How to Connect On-Prem Leaders to Cribl.Cloud. These instructions also explain how to:

- Optionally set up environment variables at the time you create the Connected Environment.
- Use a command-line script to set up a Connected Environment if needed.

Ensure that you also follow the instructions to Enable the On-Prem Leader Connection to Cribl.Cloud.

On-Prem Edge Nodes: Configure HTTP or TCP Destination

On the on-prem deployment, select a Edge Node and configure a new or existing Cribl HTTP or Cribl TCP Destination. These Destinations enable internal data transfer between Cribl nodes to prevent data transfer billing costs.

To configure an HTTP or TCP Destination:

1. Follow the general configuration instructions on the Cribl HTTP or Cribl TCP Destination documentation.
2. In the **Auth Token** section of the configuration modal, select **Auth Token** to set up a shared secret between this Destination and the Cribl.Cloud Source:
    - **Token secret (text secret):** Select the name of the secret that you have created previously or create a new secret using the **Create** button. See Create and Manage Secrets for specific instructions about creating secrets.
    - **Enable token:** Toggle the token on to enable it.
    - **Description:** Optionally, enter a description summarizing the purpose of the secret.



If you don't see the **Auth Token** section of the Destination, it means you have not set up a Connected Environment for your on-prem Leader.

3. Commit and deploy the changes.

You can add multiple tokens if you need to rotate auth tokens.

Cribl.Cloud Edge Nodes: Configure HTTP or TCP Source

On the Cribl.Cloud deployment, select a Edge Node and configure a new or existing Cribl HTTP or Cribl TCP Source. These Sources enable internal data transfer between Cribl nodes to prevent data transfer billing costs.

To configure an HTTP or TCP Source, follow the same steps listed in the previous section about Configuring HTTP or TCP Destination for the on-prem Edge Nodes.

When you select or create the secret from the **Token secret (text secret)** menu, this secret must match the value of the secret used on the on-prem Cribl HTTP or Cribl TCP Destination.

Commit and deploy after making these configuration changes.

Validate the Data Flow

As a final step, confirm that the data is properly flowing from the on-prem Edge Nodes to the Cribl.Cloud Edge Nodes. A few methods for confirming data flow include:

- Capture sample data on your Cribl HTTP or TCP Source on your Cribl.Cloud Edge Nodes and verify it looks correct. See Add Sample Data for specific instructions.
- In the configuration modal for your Cribl HTTP or TCP Source on your Cribl.Cloud Edge Nodes, select the **Charts** tab to view data events coming in to verify you see activity. You can also compare this to the outgoing data on the Cribl HTTP or TCP Destination on your on-prem Edge Nodes.

7.4. Tutorials

Follow these tutorials to get a one-stop overview of the interface and data flow functionalities of Cribl Edge.

- Getting Started Tutorial walks you through basic Cribl Edge functionalities, showing you how to capture sample data and send it to Cribl Stream.
- Configuring Cribl Edge offers practical guidance on configuring your new deployment to efficiently collect, process, and forward your Cribl Edge data.
- Forward Custom Data with Cribl Edge Tags shows how to add unique identifiers or metadata to each event for better control of data at scale.
- Monitoring your Infrastructure with Cribl Edge presents how to monitor infrastructure by collecting, processing, and forwarding data with low overhead cost and resources.
- Monitor and Troubleshoot Cribl Edge explores key metrics to monitor for optimal Cribl Edge operation, discusses how to gather this information, and provides effective monitoring strategies and general troubleshooting guidelines.

7.4.1. Getting Started Tutorial

Learn basic Cribl Edge functionalities.

This tutorial walks you through basic Cribl Edge functionalities, showing you how to capture sample data and send it to Cribl Stream.

To complete this tutorial you'll need two instances: one running Cribl Edge, and one distributed Cribl Stream. If you do not have any yet, refer to Getting Started Guide for the former, and Stream's Distributed Quick Start for the latter.

You do not need connections to any external services, and a free Cribl license will be enough to go through all the steps.

Add the File Monitor Source

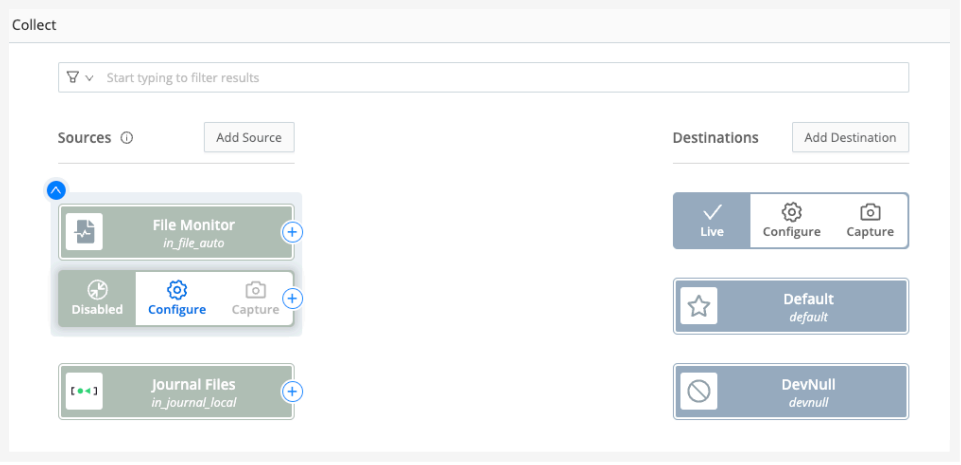
Each Cribl Edge Source represents a data input. Options include System Metrics, File Monitor, Kubernetes Metrics, Logs, and Events, as well as Prometheus Edge Scraper, and others.

For this tutorial, we'll use the File Monitor Source which generates events from text files or log files.

1. In Cribl Edge, select **Manage** > **Collect**. You can use the QuickConnect screen to configure how data flows between Sources and Destinations.

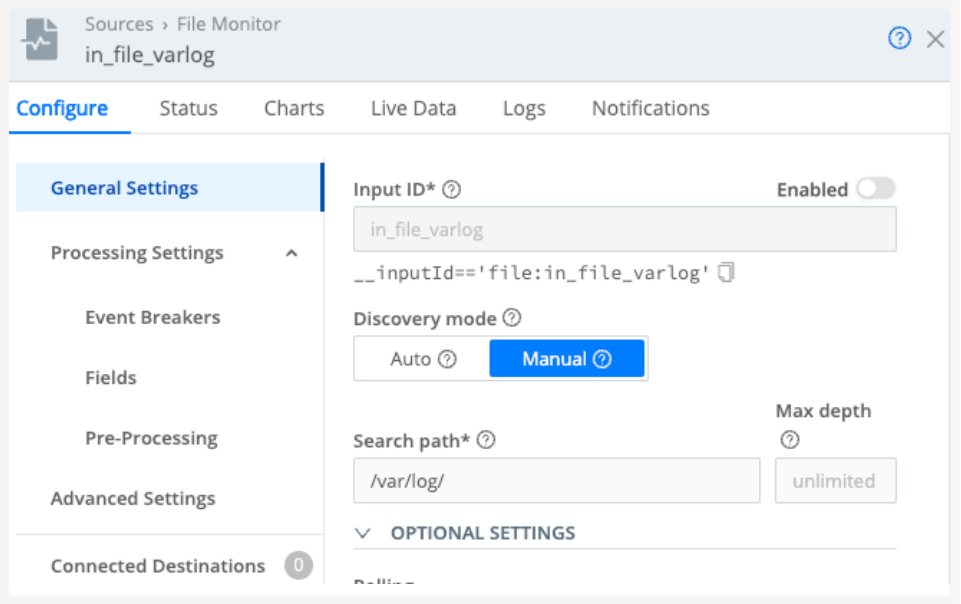
   If you're on Cribl.Cloud or any other Distributed mode, first select the `default_fleet` in its tile to select it.

2. From the list of Sources, expand the File Monitor tile and select **Configure** for **in_file_varlog**.



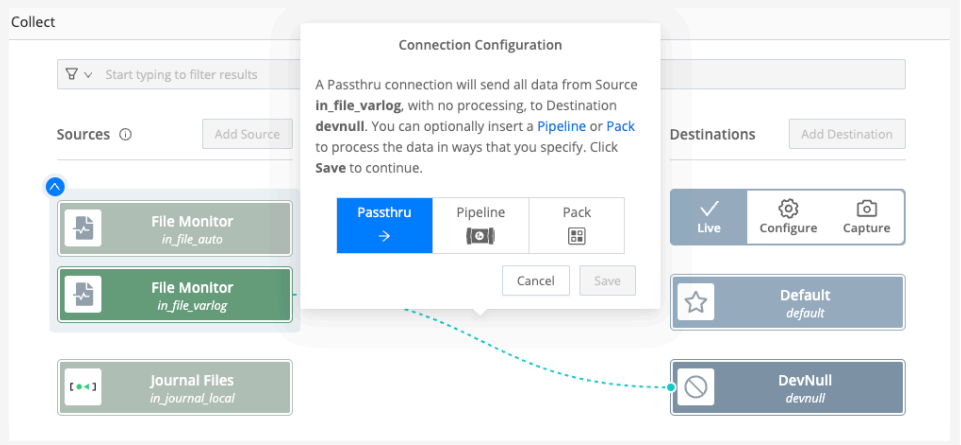
QuickConnect UI with the built-in File Monitor Sources.

3. When set to a Manual discovery mode, the **Search path** field in File Monitor specifies the path that the Source should look through to discover files. **in_file_varlog** contains the `/var/log/` path, which works for this tutorial.



The default settings for the Source are good enough for this tutorial.

4. Exit the modal. Now let's activate the Source.
5. In the QuickConnect screen, drag the + icon from the File Monitor tile to the **DevNull** Destination on the right.
6. In the **Connection Configuration** modal, select **Passthru**. This indicates that the data should pass through without any modification.



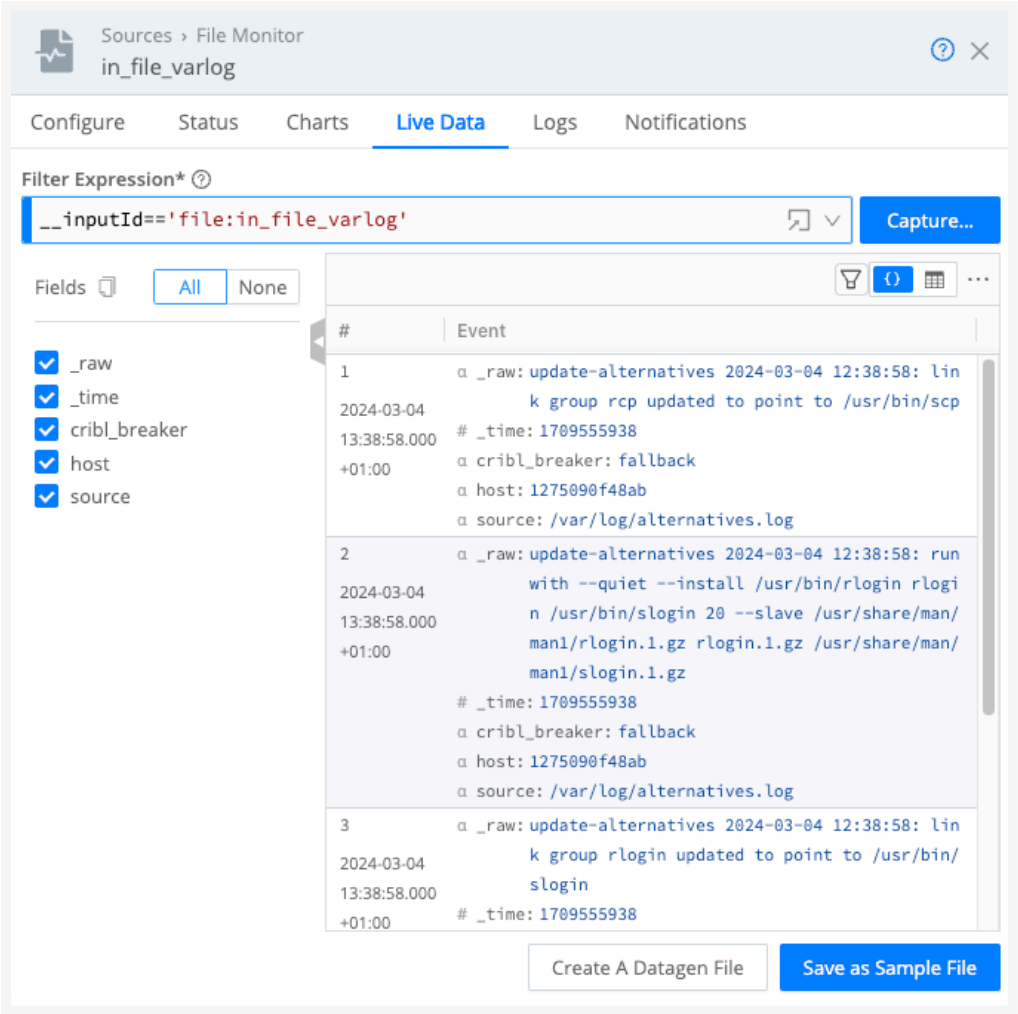
QuickConnect UI with a new connection between File Monitor and DevNull.

7. Select **Save**.

If you're on Cribl.Cloud or any other Distributed mode, select **Commit** before proceeding. Then, in the resulting dialog, type a commit message (for example, "Added Source") and select **Commit and Deploy**

> to confirm. You'll see a **Commit successful** message.

Now select **Capture** in the **FileMonitor** Source to go to the **Live Data** tab. You'll see data flowing from the Source.



Data from the File Monitor is already flowing.

Hover over the **DevNull** Destination to see a shortcut menu. You can select **Capture** to jump to a preview of data flowing into this Destination.

If you do not see any events, the reason may be that no file changes were registered in this period. In that case you can select **Capture...** and increase the **Capture Time (sec)** value to extend the time window for catching events, then confirm with **Save**.

Add a Cribl HTTP Destination

Each Cribl Edge Destination represents a data output. Options include Cribl HTTP, Cribl TCP, Kafka, Kinesis, InfluxDB, Grafana Cloud, TCP JSON, Splunk, and others.

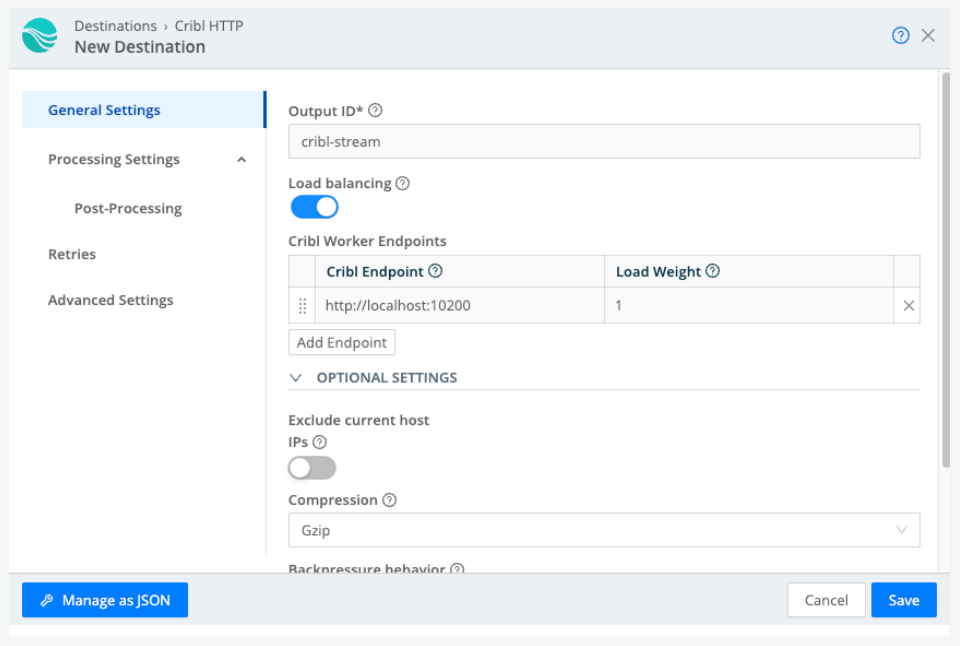
So far you've temporarily used the **DevNull** Destination. Now, let's set up an actual Destination where you'll sent the data to a separate Cribl Stream instance.

We'll use the Cribl HTTP Destination, which enables sending data between different Nodes, and connect it to the Cribl HTTP Source in Stream to receive the data.

1. From Cribl Edge's top menu, select **More** > **Destinations**.

   If you're on Cribl.Cloud or any other Distributed mode, first select the `default_fleet` in its tile.

2. From the **Manage Destinations** page's tiles or left menu, select **Cribl HTTP**. (You can use the search box to jump to the **Cribl HTTP** tile.)
3. Select **Add Destination** to enter a **New Destination** modal.
4. Provide your Destination's **Output ID** (for example, `cribl-stream`).
5. In **Cribl Worker endpoints** you will provide the endpoint of the Cribl Stream instance you want to send the data to.
   Provide the address for your Cribl Stream instance appended with the `10200` port, for example: `http://localhost:10200/`.



Cribl HTTP Destination modal with endpoint configured.

6. Confirm with **Save**.

If you're on Cribl.Cloud or any other Distributed mode, select **Commit** at Cribl Edge's upper right before proceeding. Then, in the resulting dialog, type a commit message (for example, "Added Destination") and select **Commit and Deploy** to confirm. You'll see a **Commit successful** message.

Receive Data

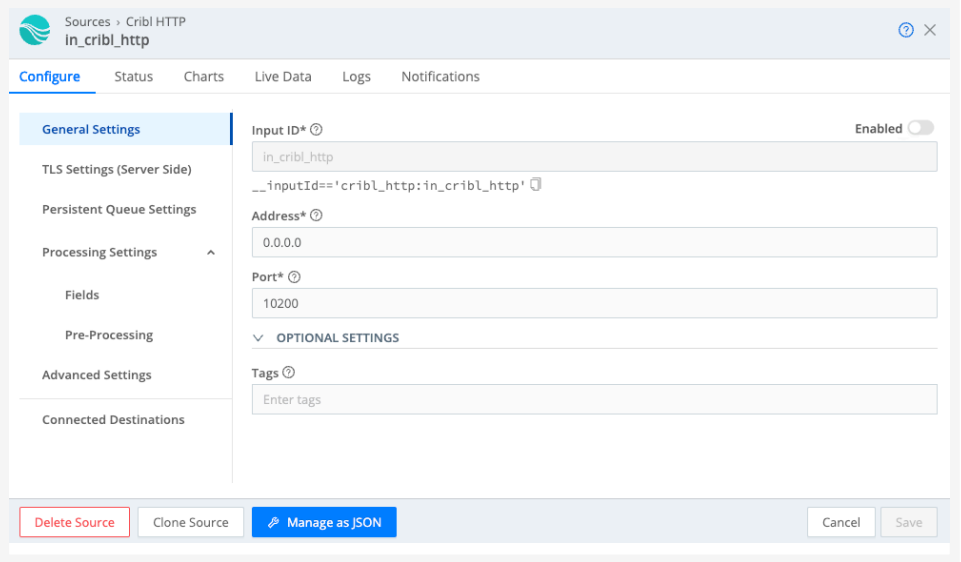
Now your Cribl Edge is ready to send the data, but Stream is not yet configured to accept it. Let's quickly jump to it to finish the preparations.

1. Access your Cribl Stream's UI, and select the `default` (or any other) Worker Group.
2. In the top menu, select **Data** > **Sources**.
3. Search for **Cribl HTTP** and select it. You will see a built-in `in_cribl_http` Source that may be disabled by default.



If you cannot find the **Cribl HTTP** Source, it means your Stream is working in a single-instance mode.

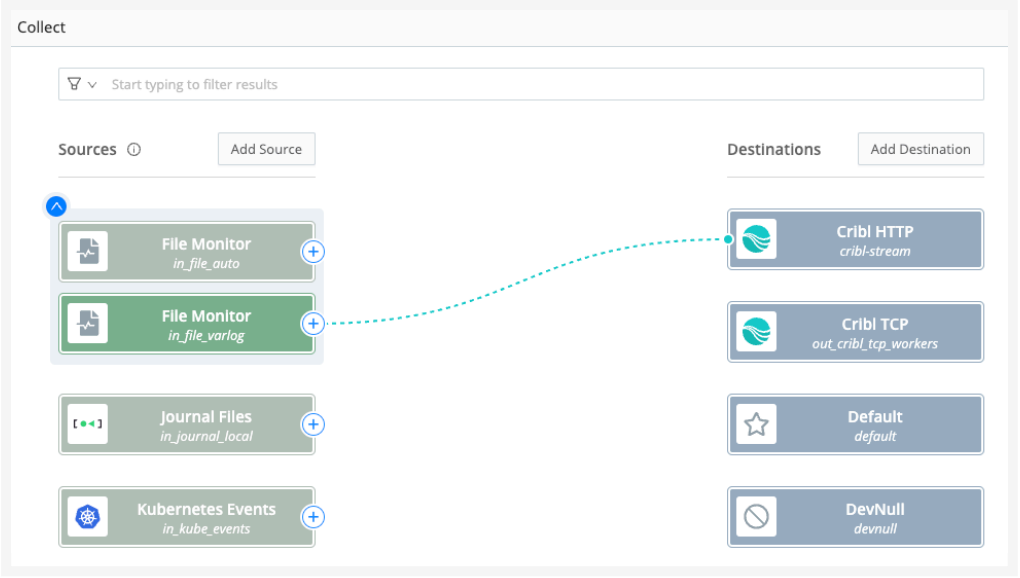
4. In the Source's configuration modal, make sure that the **Address** Field points to `0.0.0.0`, which is a catch-all for all addresses, and that **Port** is the same you configured for the Destination in Edge (using the example above, `10200`).



This Source will pick up events sent by Cribl Edge.

5. Enable the Source and exit the modal.
6. Select **Commit** at Cribl Stream's upper right. In the resulting dialog, type a commit message and select **Commit and Deploy** to confirm. You'll see a **Commit successful** message.

Go back to your Cribl Edge UI and on the **Collect** screen switch **File Monitor** to send to **Cribl HTTP** instead of **DevNull**



The data now flows where it should: to the Cribl HTTP Destination.

If you're on Cribl.Cloud or any other Distributed mode, select **Commit** at Cribl Edge's upper right before proceeding. Then, in the resulting dialog, type a commit message (for example, "Added Destination") and select **Commit and Deploy** to confirm. You'll see a **Commit successful** message.

Wait a few moments for the data to start flowing, and then in Cribl Stream select the `in_cribl_http` Source's **Live** button. After a few seconds you should see a sample of data flowing into Cribl Stream from your Edge instance's File Monitor.

The data you are receiving is now ready for further reduction and transformation using Cribl Stream's Routes, Pipelines and Function. To get a short introduction to Stream's capabilities in working with data, you can refer to its Getting Started Guide.

Summary

In this short tutorial - with no hit to your cloud-services charges - you've built a simple but complete Cribl Edge instance, exercising all of its basic components:

- Configured a Source to monitor files in selected disk location.
- Configured a Destination to send data to a Stream instance.
- Configured Stream to receive data from Edge.

Cleaning Up

You can finish up the tutorial by cleaning up after yourself. You've still got the Cribl Edge server running, with files still being monitored. If you'd like to shut it down for now, in reverse order:

1. Go to **Manage** > **More** > **Sources** > **File Monitor**.
2. Toggle **in_file_varlog** off.
3. In your terminal's `$CRIBL_HOME/bin` directory, shut down the server with: `./cribl stop`

Next Steps

Interested in guided walk-throughs of more-advanced Cribl Edge features? We suggest that you next check out these further resources.

- Cribl Edge Sandbox: Work through general and specific scenarios in a free, hosted environment, with terminal access and real data inputs and outputs.
- Explore Cribl Edge: Learn more about the various capabilities of Cribl Edge by taking a tour through its UI

7.4.2. Configure Cribl Edge

Look through a checklist of steps to configure a complete Cribl Edge deployment.

This guide offers practical guidance on configuring your new deployment to efficiently collect, process, and forward your Cribl Edge data. We understand that every environment and requirement is unique, so we are offering flexible recommendations to help you build a tailored solution. Topics include:

- **Customize your Cribl Edge environment** by configuring Global Settings, certificates, and metrics limits.
- **Design effective Fleets and explore Fleet mapping** to group together Edge Nodes based on the data you are collecting.
- **Configure essential Sources and Destinations** to meet your data collection needs, including considerations for specific Sources and Destinations.
- **Determine the best approach for routing data**, by using either Routes or QuickConnect.
- **Optimize data flow** with Pipelines.
- **Use Pre-built Packs** to streamline configuration and accelerate deployment.

Customize Your Cribl Edge Environment

Begin by customizing your Cribl Edge instance in Global Settings. This includes:

- **Branding**: Adjust the UI display, including custom banners and login page, to match your organization's branding.
- **Security**: Implement SSL/TLS certificates to secure communication between the Leader and its Edge Nodes. For details, see the Secure Leader/Edge Nodes Communication guide. In Cribl.Cloud, SSL/TLS certificates automatically secure communication between the Leader and Nodes, simplifying deployment and ensuring data protection.
- **Optimize Cribl Metrics on the Leader and Edge Nodes, respectively**: Review performance considerations and set metrics limits to ensure optimal performance and resource utilization. See Controlling Metrics in Cribl Edge for a description of each option.

Effective Fleet Design

Cribl Edge enables you to create flexible groups of Edge Nodes, called Fleets and Subfleets. Design your Fleets **based on the specific data being collected** by each Edge Node.

When migrating from other agents to Cribl Edge, it's essential to avoid the common pitfalls of configuring a single Fleet for all your Edge Nodes, or creating Fleets to replicate other agent configurations. These

approaches can lead to inefficient data collection and management.

By instead creating targeted Fleets, you can optimize your data collection, processing, and flow. The Leader's platform-agnostic nature ensures seamless integration of your Fleets. Use Fleet Mapping, a feature on the Leader Node, to streamline the management of heterogeneous Edge Nodes by organizing them into Fleets. For detailed guidance beyond the scope of this guide, see Design Fleet Hierarchy.



Configuring multiple Fleets is available with an Enterprise plan. For details, go to the Edge - Cloud Ingest guide.

Fleet Mapping

Cribl Edge Mapping Rulesets enable you to effectively organize and classify Edge Nodes based on various criteria, such as tags, operating systems, environment variables, and more. This helps you create granular Fleet and Subfleet structures for optimized management and scaling.

Common mapping criteria include the following:

- **Operating System (OS):** Use the platform field to filter by OS - for example: `platform == 'linux'`.
- **Environment Variables:** Leverage environment variables to identify specific environments or applications - for example: `env['ENVIRONMENT'] == 'production'`.
- **Tags:** Utilize tags to categorize Nodes based on custom attributes - for example: `instance.tags.some(tag => tag.key == 'Environment' && tag.value == 'Production')`.
- **Cloud-Specific Attributes:** For cloud deployments, use attributes like instance IDs, VPC IDs, or bucket regions.
- **Server Name:** Use the `host` field to filter by Server name, if you have well-defined server naming conventions in your company.

For example, you can create a Fleet named `Prod-Windows-us-east` containing all Windows Nodes in a specific availability zone within your production environment. Use a filter like `platform==="windows" && aws.region==="us-east-2"` and assign it to your Fleet.

Refer to Create a Mapping Ruleset for details on how to map Edge Nodes.

Configure Essential Sources and Destinations

With your Fleets and Subfleets defined, it's time to configure Sources to collect the data you need, and to configure Destinations to receive the processed data. Sources are responsible for ingesting data from various systems, while Destinations send the processed data to your desired storage or analytics platforms.

Configure Sources to Collect Data

Choose appropriate Sources based on your data collection needs. Commonly used Sources on Cribl Edge include:

Internal/System Sources

- System State
- Exec

Kubernetes-Specific Sources

- Kubernetes Metrics
- Kubernetes Logs

Linux Sources

- System Metrics
- Syslog

Windows Sources

- Windows Events Logs
- Windows Metrics

Platform-agnostic

- File Monitor
- TCP



While Cribl Edge can receive inbound data via Syslog or TCP, we generally discourage using Edge to ingest large volumes of data from external sources, due to its single-process nature and focus on running as an agent. Excessive network traffic can lead to performance bottlenecks or data loss.

This applies to any external data feed that is sent to the node hosting a Cribl Edge instance. If you require more performance, or need to scale your data collection, consider using Cribl Stream instead.

> Cribl Stream is designed for larger-scale data ingestion and processing, and our Sizing and Scaling guidance can help you determine the optimal configuration for your specific needs.

Considerations for the File Monitor Source

The File Monitor Source in Cribl Edge is a powerful tool for collecting file-based data, but it can also be challenging to configure effectively.

To optimize Cribl Edge's performance when monitoring systems with a large number of files (tens of thousands or more), consider the following guidelines for each phase:

- **Discovery**: Involves scanning files to obtain their modification times, and can be time-consuming for extensive file systems. If you already know the specific files to monitor, you can optimize discovery speed by setting the **Discovery mode** to **Manual.** This bypasses the automatic file scanning process, reducing performance overhead and focusing resources on the files of interest. Limiting the scope of the File Monitor by using **Manual** mode's **Search path** and **Max depth** settings can significantly improve discovery speed.
- **File Monitoring**: Enhance the efficiency of the file monitoring phase, by filtering out older, irrelevant files. Adjust the **Maximum age duration** and **Check file modification times** settings.

Discovery Phase

To use the File Monitor Source effectively, it's important to understand the interplay among the **Search path**, **Max depth**, and **Filename allowlist** settings. The following settings only appear when the **Discovery mode** is set to `Manual`:

- **Search path** specifies the root directory from which the File Monitor will start searching for files.
- **Max depth** defines how deep the File Monitor will recurse into subdirectories. A value of `0` limits the search to the specified search path only, excluding subdirectories.
- **Filename allowlist** is a list of file patterns to include in the monitoring. The allowlist field accepts only file patterns, not directory paths.

The File Monitor Source first identifies potential files based on the **Search path** and **Max depth** settings. Then, it applies the **Filename allowlist** patterns to filter the list of files. Only files matching both the **Search path** and **Filename allowlist** criteria will be monitored.

Some guidelines to keep in mind:

- Specify the full path in the **Filename allowlist**, for precise control over which files are monitored. For example: `/var/log/messages`. To optimize performance, avoid using excessive wildcards in the Cribl Search path, as this can significantly increase the number of files scanned.

- Use `max depth=0` for specific directories: If you want to monitor files only within a specific directory, set **Max depth** to `0` and specify the directory in the search path.
- Take advantage of the Explore UI: The Fleet's **Explore** > **Files** tab can help you visualize the file structure and test different allowlist patterns before deployment.

For details and examples, refer to Using Allowlists Effectively.



Cribl Edge does not support **Auto** mode on Windows or macOS hosts. While the option will still appear in the UI for Edge Fleets that include Windows or macOS hosts, selecting it will have no effect. This is because the Leader does not adjust the user interface based on the operating systems of the connected clients.

Monitoring Phase

The **Maximum age duration** and **Check file modification times** settings apply to both **Manual** and **Auto** discovery modes in the File Monitor.

Some guidelines to consider when adjusting these settings are:

- **Setting Check file modification times alone is not effective**: If you only enable **Check file modification times** without specifying an **Age duration limit,** the File Monitor will still process all files that match the filter criteria. To effectively filter files based on age, you must specify both settings.
- **Setting age duration alone reads all files:** Without enabling **Check file modification times**, the File Monitor will read all files that match the specified **Filename allowlist**. No age criteria will be applied to the entire file. However, the **Maximum age duration** will still be used to filter events within the files that have a timestamp older than the specified duration.
- **Setting both ensures efficient filtering:** To achieve optimal filtering, set both **Maximum age duration** and **Check file modification times**. This configuration allows the File Monitor to skip files that are older than the specified duration, based on their modification time. And within the remaining files, it will skip events that are older than the specified duration.

Considerations for the Kubernetes Logs Source

Cribl Edge leverages the Kubernetes API to interact with the cluster and collect logs. When deployed using Helm charts, Edge operates as a DaemonSet within a Kubernetes cluster. This deployment model ensures that a copy of Edge runs on each Kubernetes Node, enabling efficient log collection from that specific Node using the Kubernetes API. Here are a few considerations to optimize your Kubernetes Logs Source, including:

- **Selective Data Collection**: Use filter expressions to match specific patterns in log messages, labels, or namespaces. The filter expressions are applied to all Edge Nodes, so carefully configure these expressions to match your specific requirements. See Filter Expression for examples.

- **Adjust Log Retention and Rotation Settings**: For high-volume applications, log rotation can occur frequently. To ensure consistent data collection, configure cluster variables to adjust log rotations and retention policies in your Kubernetes deployment.

Considerations for the Windows Events Logs Source

Our Windows Event Logs Source directly interacts with Windows operating systems, eliminating the need for helper tools like PowerShell. This results in:

- **Faster collection**: Direct polling ensures more efficient data retrieval.
- **Flexible data formats**: Choose between XML or JSON to suit your specific needs. XML offers faster data collection than JSON, but this efficiency can sometimes cause some loss of event detail.
- **Message Rendering:** Choose to render message strings in the events for context.

Data Routing from Edge Nodes

Edge Nodes can be configured to route collected data in several ways:

- **To Cribl Stream:** For further processing and scalable distribution.
- **Directly to Destinations:** Such as cheap storage, SIEMs, or other security and IT tools.
- **Selectively:** Combining direct delivery plus routing to Cribl Stream, based on use cases.

Most customers prefer to offload as much workload from their agents as possible. To achieve this, use Cribl Edge to:

1. Collect data locally.
2. Filter out unwanted data, using the Drop Function.
3. Route the remaining data to Cribl Stream Workers for further processing and delivery to your desired Destinations.

Cribl Edge supports a variety of Destinations for your data. Some of the most common options are:

- Cribl TCP
- Cribl HTTP
- S3
- Splunk

Considerations for Cribl TCP vs. Cribl HTTP Destinations

Cribl HTTP is generally recommended over **Cribl TCP** for data transmission between Edge and Worker Nodes connected to the same Leader, especially in environments with proxies or firewalls. Cribl HTTP provides greater flexibility and ease of integration with various systems.

The choice between the two ultimately depends on specific use-case requirements, such as the need for low latency, high throughput, or compatibility with external systems. To optimize performance, consider advanced configuration options like throttling, load balancing, and connection pooling.

QuickConnect vs. Routes in Cribl Edge

QuickConnect and **Data Routes** are both interfaces for managing data flow in Cribl Edge, but they serve distinct purposes, based on the complexity of your routing needs.

QuickConnect is ideal for simple, straightforward scenarios with a given Source sending all data to Destination(s). It offers ease of use and speedy configuration.

Data Routes, on the other hand, are better suited for complex data routing involving multiple Sources, Destinations, and conditional processing. They provide flexibility, advanced logic, and scalability for handling intricate data flows.

Choose QuickConnect for rapid, minimal configuration setups, and opt for Routes when your data requires more sophisticated routing and processing. For details, see Routes.

Processing Cribl Edge Data with Pipelines

Once you have collected your data, you have options to transform it before sending it to your Destinations.

You can transform data within pre-processing, processing, or post-processing Pipelines. In this section, we will highlight a few Functions/Pipelines to help you transform and optimize your data.



Limit Pipelines on Edge Nodes to basic Cribl Functions. Avoid complex operations, to prevent bottlenecks and ensure optimal performance. You can check the Pipeline Profiler to access statistics on each Pipeline's performance.

Drop Events to Lower Ingest Cost

By using the Drop Function, you can effectively eliminate data from your ingestion pipeline without incurring additional costs.

There are few ways you can take advantage of this Function:

- Use pre-processing Pipelines to filter and drop specific events, based on criteria like event codes.

- Use processing Pipelines to drop data before cloning events to send them to multiple Destinations. This reduces data volume.
- Safely experiment with configurations, without incurring costs, by routing data to the DevNull Destination.

For details, refer to the Get More Out of Cribl Edge by Dropping Events blog post.



The number of dropped events that lower the ingestion cost is calculated based on average event size. This means it may not represent the exact number of bytes you drop.

Enrich Events by Adding Metadata

To enrich and refine your data within Cribl Edge, effectively use the **Eval** Function's **Add Field** option. This feature enables you to dynamically create new fields based on existing data elements - such as tags from mapping rules, or event metadata. By promoting tags and metadata to fields, you can enhance data analysis and manipulation capabilities.

For instance, a File Monitor might be collecting IIS logs from a Windows server. To ensure that these events are stored correctly in your final analysis system, you could specify a value of `index = 'IIS'`. This creates a new field named `index` with the value `IIS`, providing essential context for your data.

For details, see Host Info.

Reduce Your Data with Lookups

Use lookup tables on Edge Nodes to filter out events that meet predefined conditions. This can significantly improve processing efficiency and reduce costs.

For more complex lookup scenarios, consider performing them on Cribl Stream instead of Cribl Edge, to offload the processing burden. This approach enables you to leverage the capabilities of Cribl Stream for advanced data manipulation and analysis, while maintaining optimal performance on Edge Nodes.

For details, refer to the Lookup topic.

Transform Logs into Metrics

Aggregating individual log events into metrics reduces throughput volume, reducing your infrastructure requirements and optimizing your data ingest utilization on downstream services.

For details, refer to our Working with Metrics sandbox, which walks you through extracting metrics from logs and publishing them.

Packs for Cribl Edge

Cribl Edge supports various Packs that can enhance Edge functionality and provide pre-built solutions for common use cases. While the specific Packs available can vary, and may be updated over time, here are some example Packs that are often used in conjunction with Cribl Edge:

- Cribl Edge for Windows
- Cribl Edge Windows for Common Schema
- Microservices Logs

When selecting Packs, prioritize features that align with your specific needs, while considering the potential

7.4.3. Forward Custom Data With Cribl Edge Tags

Add unique identifiers to events forwarded from Cribl Edge by using tags.

Organizations often need to add unique identifiers or metadata to each event forwarded from Cribl Edge to supported Destinations. Whether you're managing a small deployment or thousands of nodes, incorporating Edge tags into your events provides the context needed to make sense of your data at scale. These identifiers can help:

- Track the source of events across a large enterprise
- Associate events with specific departments, teams, or business units
- Maintain custom organizational taxonomy in log data
- Enable better filtering and searching in the Destination systems

This guide will show you how to set tags during Cribl Edge Installations and use the Preprocessing Pipeline to forward custom data points to a Destination.

Embed Source Context with Cribl Edge Tags

In this guide, we will follow Data Detective Dave, an operator with a knack for getting to the bottom of data origins. His current case involves enriching data streaming from all Development servers within the Engineering Organization before it's dispatched to a central analytics platform. Dave needs to ensure every piece of data clearly shouts its server role and department.

Here are the steps Data Detective Dave will follow:

1. **Set meaningful tags while installing Cribl Edge:** Dave's first move is to tag each Development server right at the Source, while installing Cribl Edge. He'll use the installation script to attach labels `development` and `engineering` – ensuring each server wears its identity from the get-go.
2. **Gather the metadata evidence:** Cribl Edge, like a good informant, automatically collects metadata, including the tags Dave just applied. He'll double-check that the `cribl` metadata option is active in the Fleet Settings, confirming that this crucial data is being gathered.
3. **Set Up the "Interrogation Room" (Preprocessing Pipeline):** Dave will create a new Pipeline in Cribl Edge, which he'll cleverly name `host_tags`. This will be his interrogation room where the data will be questioned and its secrets revealed (in this case, the tags).
4. **Use the "Eval-uator" to surface the clues (Eval Function):** Inside the `host_tags` Pipeline, Dave will employ an **Eval** Function – his "Eval-uator." This function will dig into the `__metadata.cribl.tags` array, where the installation tags are stashed. It will then create new, easy-to-read fields in the data: `role` and `department`, clearly displaying the tag values.

5. **Apply the interrogation to the Source:** Next, Dave will apply the `host_tags` Pipeline to the specific Source(s) feeding data from the Development servers. This ensures that all incoming data undergoes interrogation and reveals its tags.
6. **Verify the findings with Live Capture:** To ensure his "Eval-uator" is working correctly, Dave will use the **Live Data** capture feature on the configured Source. By examining the data before it's routed, he can confirm that the `role` and `department` fields are present and accurate - case closed (almost)!
7. **Ensure the Destination doesn't lose the evidence:** Finally, Dave will make sure the central analytics platform – the Destination – is set up to receive and hold onto these newly revealed role and department fields. He might even need to set up specific routes to handle this tagged data appropriately.

If you'd like to follow along Data Detective Dave's journey, we outlined the detailed steps for you below.

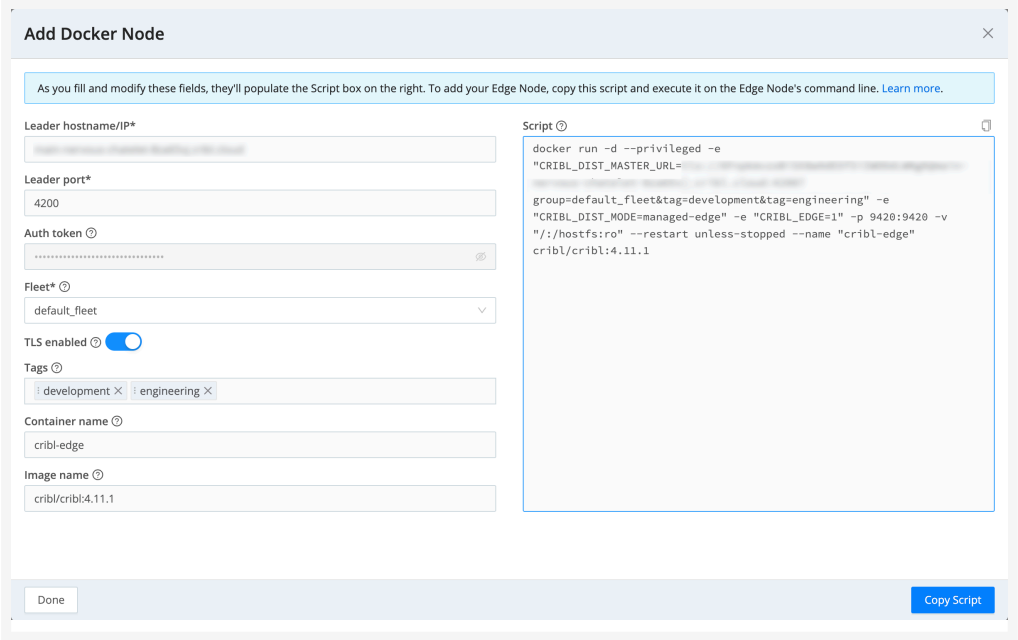
Set Tags During Edge Installation

When installing Cribl Edge, you can set tags during the installation. Regardless of your deployment environment (Docker, Linux, Windows, macOS, or Kubernetes), the Bootstrap script via the UI, gives you an option to add tags. These tags help identify and categorize your Edge Nodes such as environment type, geographic location, or server role.

As you modify the fields within the script interface, it updates in real-time, reflecting all the configuration choices you make on that screen, including any tags you've added. The result is a new Edge Node configured with the specific options you've selected.

In this example, we're setting two tags:

- Server Role: `development`
- Department: `engineering`



You can also add tags after you install the Edge Nodes by teleporting directly into the Node. For details, see Add Tags to Edge Nodes in Teleport View.

Configure Host Info/Metadata Collection

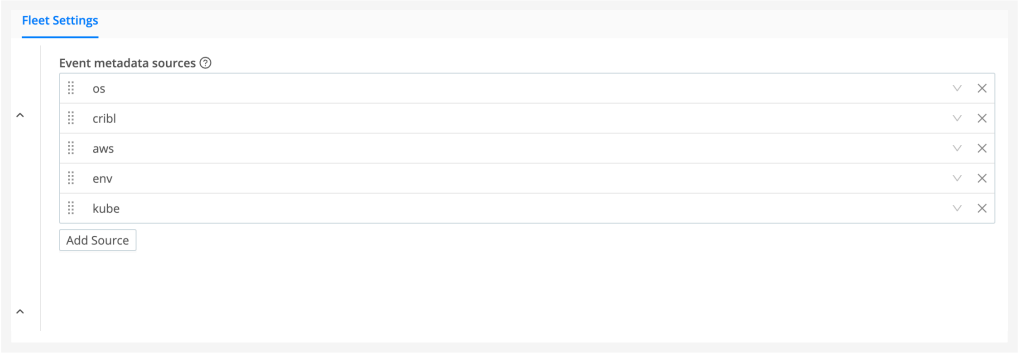
Cribl Edge allows you to configure Event metadata sources to enrich events with additional context. In Edge mode, all the Event metadata sources are enabled by default.

The Event metadata sources add a `__metadata` property to every event emitted from enabled sources. This metadata can be used to:

- Enrich events with additional context.
- Display metadata for instance exploration.

Ensure the `cribl` metadata option is enabled by checking in:

1. Navigate to **Fleet Settings** > **Limits** > Metadata.
2. Use the **Event metadata sources** drop-down or the **Add source** button to enable and select metadata sources. Ensure the `cribl` option is enabled.



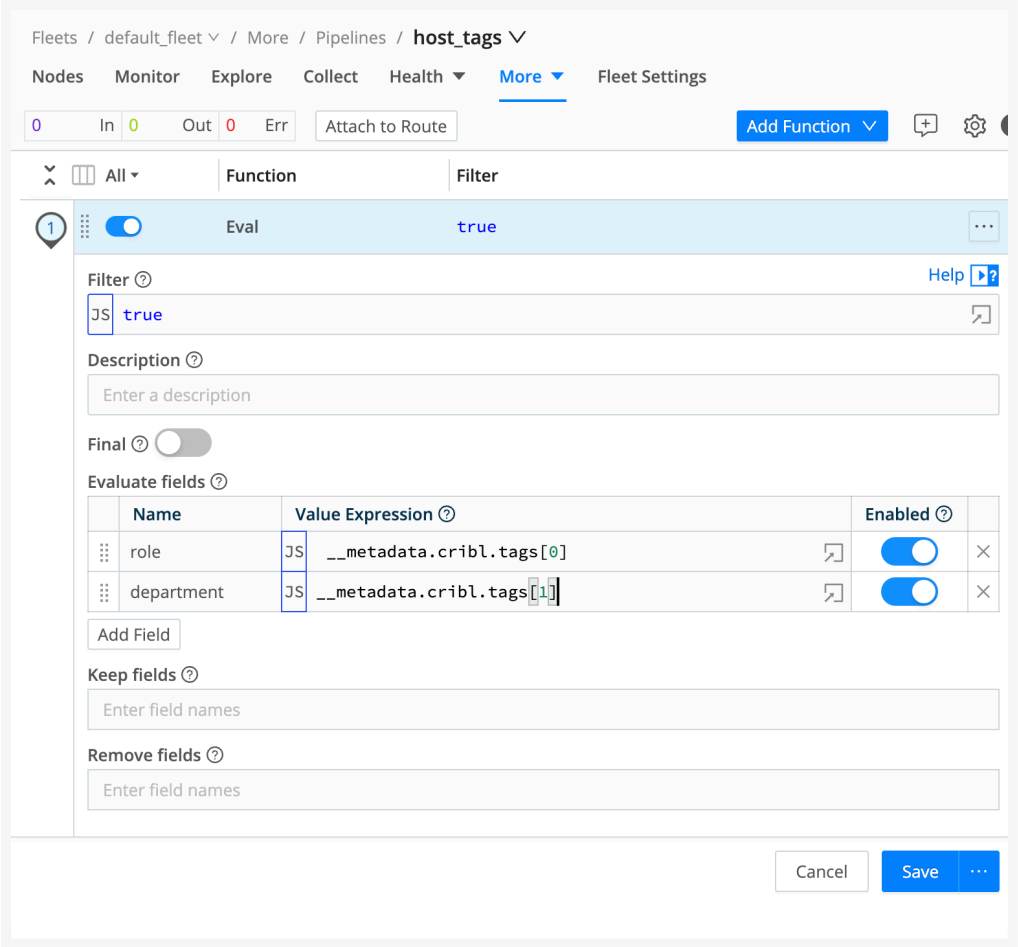
Event Metadata Sources

Create a Pre-Processing Pipeline

To add tags or other custom identifiers to events:

1. Navigate to your **Fleet** > **More** > **Pipelines** in your Cribl Edge interface.
2. Add a new Pipeline (e.g., `host_tags`). Pipelines ID can contain only letters (`A-Z`, `a-z`), numbers (`0-9`), and the characters `_` and `-`.
3. Add an **Eval** Function to the Pipeline.
4. Configure the Eval function to surface the tags by adding the following to the **Evaluate fields** section:

1. In the first row enter:
   Name: `role`
   Value Expression: `__metadata.cribl.tags[0]`
2. Select **Add Field** and enter:
   Name: `department`
   Value Expression: `__metadata.cribl.tags[1]`



Eval Function

1. Save the Pipeline.

Apply the Pre-Processing Pipeline

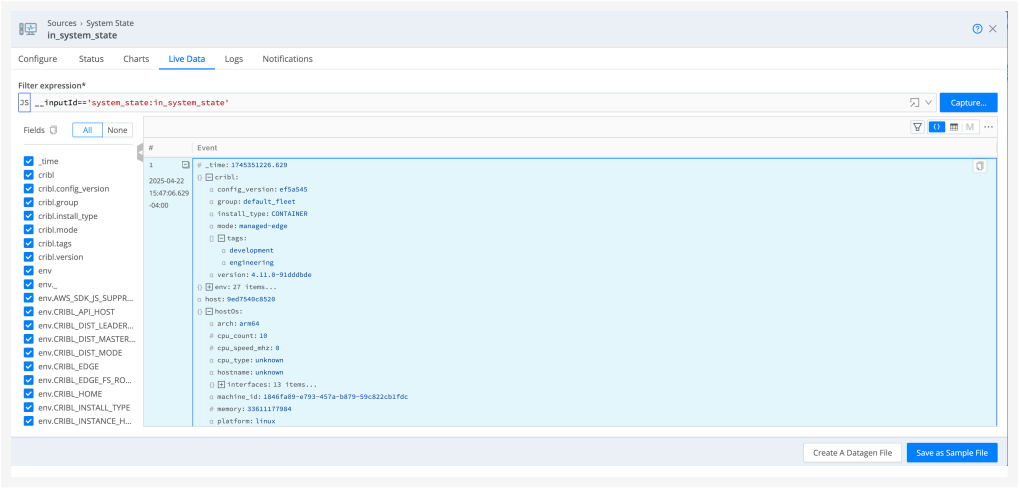
To make sure this Pipeline runs for all your data:

1. Go to **Sources > [Your Source]**.
2. In the Source configuration modal, find the **Pre-Processing Pipeline** option.
3. Select your newly created `host_tags` Pipeline.
4. Save the configuration.

Verify the Output

To verify that your custom fields are correctly added:

1. On your Source configuration modal, select **Live Data** > **Capture**.
2. In the **Capture** modal, under **Where to capture?**, select `2. Before the Routes.`
3. Verify that the `role` and `department` fields appear in the events.



Live Data

Destination Configuration

Ensure that your Destination configuration isn't dropping these custom fields:

1. Navigate to **Data** > **Destinations**.
2. Configure the Destination.
3. If needed, create a Route that specifically handles these tagged events.
4. Verify the data captured includes the additional metadata fields you configured.

7.4.4. Monitor Your Infrastructure With Cribl Edge

Monitor your infrastructure and visualize performance using Grafana.

Cribl Edge helps you monitor your sprawling infrastructure by collecting, processing, and forwarding data with low overhead cost and resources. Cribl Edge simplifies agent administration and data collection for your endpoints, and provides a number of charts, graphs, and alerts for the web when connected to supported data sources.

To enhance the performance of your environments, combine Cribl Edge with your data visualization tool of choice (this guide uses Grafana as an example). Now, you can identify and investigate issues and monitor applications faster than ever before.

Useful Metrics Dashboards

Cribl has configured a Prometheus dashboard to display system metrics, which is available in Grafana's public library. This is a relatively simple dashboard, suitable for showing basic aggregate metrics.

Another useful dashboard for displaying Linux metrics is the Node Exporter Full dashboard, commonly used with Prometheus and its Node Exporter agent. This dashboard can handle highly detailed metrics.

To display Windows metrics, consider one of Grafana's Windows dashboards.

Guide Overview

This setup guide will walk you through these configuration steps:

1. Configure your Cribl Edge Sources. Depending on your endpoints, you can use the System Metrics Source for Linux, Windows Metrics, or Kubernetes Metrics.
2. Configure the Grafana Cloud Destination to send data to Grafana Cloud.
3. Access Grafana Cloud to see the metrics and visualizations.

Configure the System Metrics Source on Cribl Edge

1. In Cribl Edge, start by configuring and enabling the applicable Source for system metrics:
    - Linux System Metrics Source
    - Windows Metrics

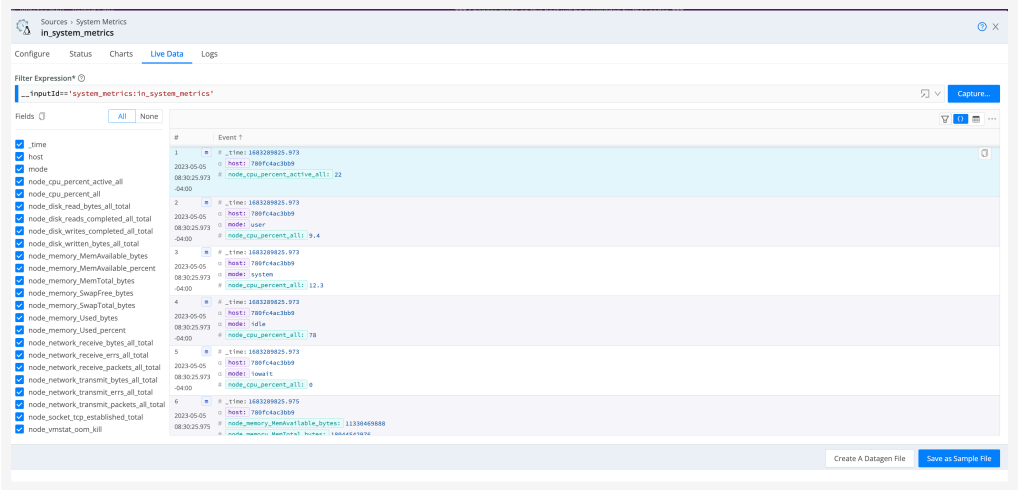
    - Kubernetes Metrics
2. Decide what level of metrics detail you want to collect and display.

In the System Metrics Source and Windows Metrics Sources, go to the **Host Metrics** and **Container Metrics** tabs:

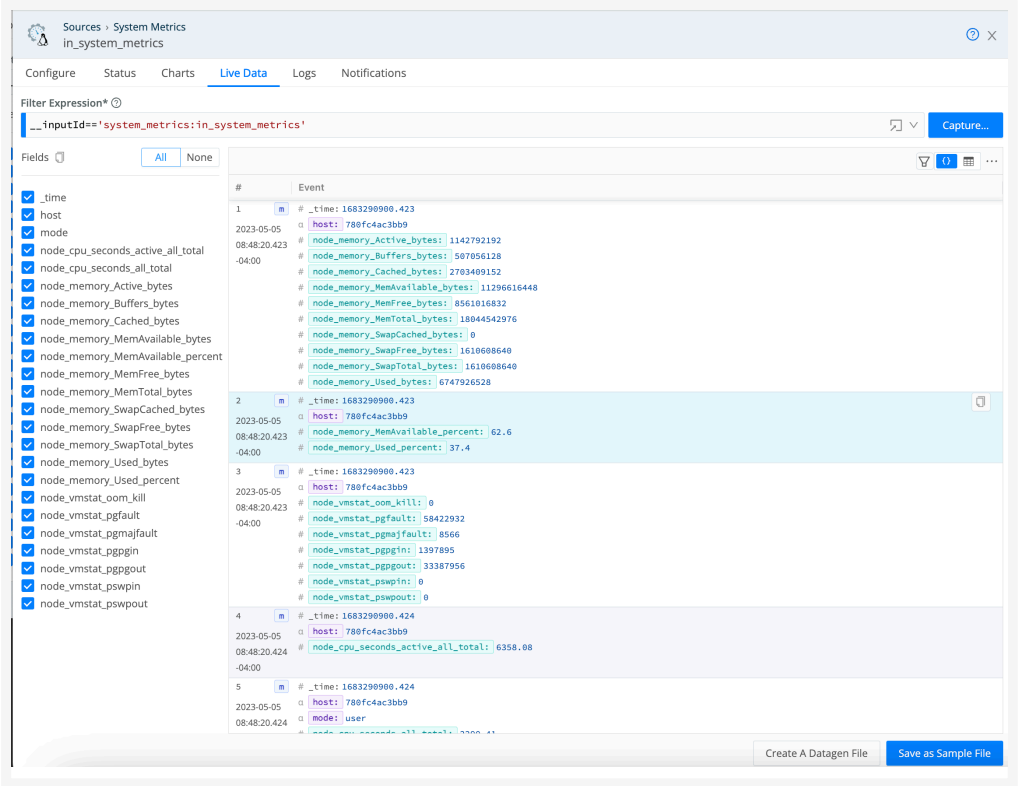
- For the relatively simple dashboard showing aggregate metrics, choose the **Basic** mode for most of your metrics.
- For the highly detailed metrics visualizations dashboard, choose the **All** mode for all of your metrics.

For the Kubernetes Metrics Source, use the **Filter Rules** to specify the Kubernetes objects that the Source should parse to generate metrics. If you specify no restrictive rules here, the Source will emit all events.

1. For all of the Sources, select **Processing Settings** > **Pre-Processing**, then select the `prometheus_metrics` **Pipeline**.
2. When you're finished, **Commit** and **Deploy** the changes. Before you move on to the next step, check the **Live Data** tab to make sure the metrics are generated.



Basic System Metrics

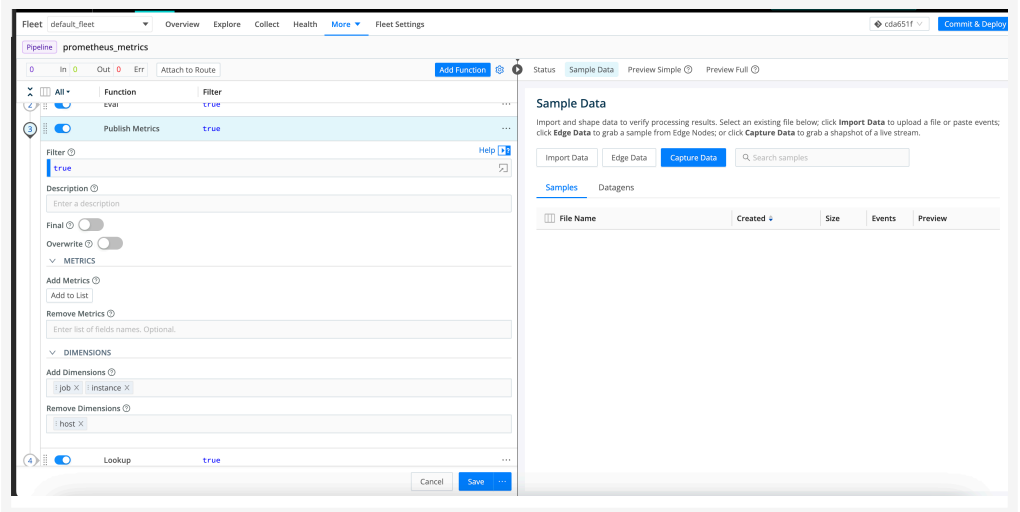


Detailed System Metrics

(Optional) Adjust the `prometheus_metrics` Pipeline

Optionally, you can adjust the `prometheus_metrics` Pipeline to include the `host` dimension, or to add/remove more dimensions.

Access the `prometheus_metrics` Pipeline by selecting **More** > **Pipelines** in the Fleet submenu.

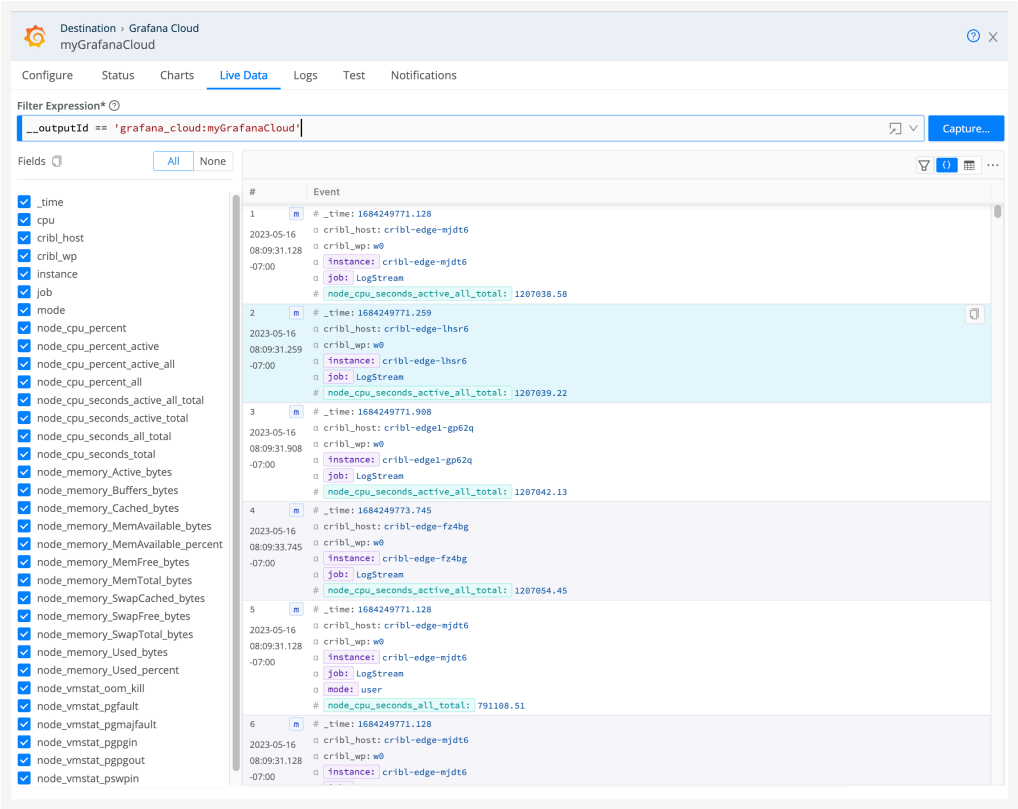


Customizing the Pipeline's dimensions

Configure the Grafana Cloud Destination

Next, head over to Grafana Cloud Destination to configure the Destination and send data to Grafana Cloud.

Preview your data on the Grafana Cloud config modal's **Live Data** tab.

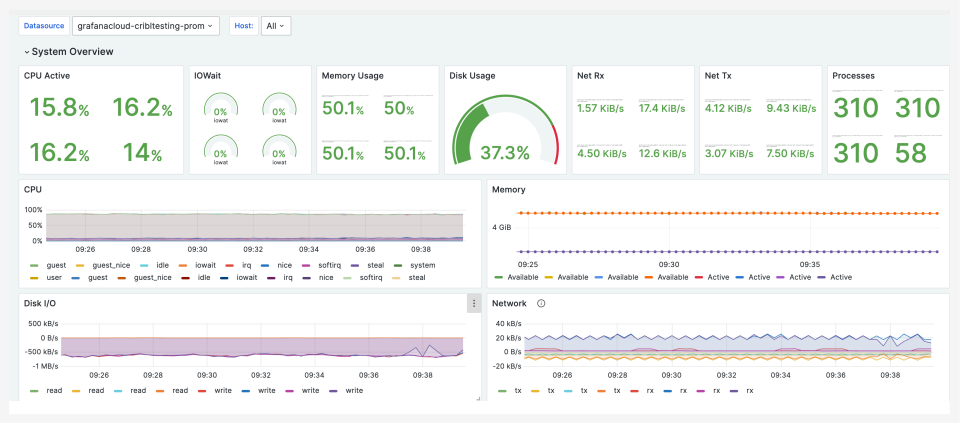


Preview Live Data

Visualize the Data in Grafana

You can now visualize metrics in Grafana Cloud. To get started:

1. Navigate to your Grafana Cloud instance, and select **Explore** on the left panel to review the Node metrics. You can select metrics and run a query to produce a chart and table of metric events, which you can add to new or existing dashboards for analysis. If expected metrics are missing, try adjusting the variables in your Grafana Cloud instance. See Grafana's Variables topic for details.
2. Use Grafana's **Filter** option to browse the dashboards for your Prometheus data. Below are sample dashboards with the basic and detailed metrics shown.
    - To show basic aggregate metrics, use the Cribl Node Observability dashboard.



Cribl Node observability

- For the OS metrics with `CPU` / `Disk` details, the basic dashboard will not suffice. Instead, we

7.4.5. Monitor And Troubleshoot Cribl Edge

Monitor and troubleshoot your Cribl Edge deployment.

Cribl Edge is a data collection agent for Windows, Linux, macOS, and Kubernetes. Even robust agents like Cribl Edge can encounter issues that disrupt data flow. To maintain optimal performance and reliability, it's essential to implement effective monitoring and troubleshooting strategies.

In this topic, we explore key metrics to monitor for optimal Cribl Edge operation, discuss how to gather this information, and provide effective monitoring strategies and general troubleshooting guidelines.

Additionally, we highlight the importance of integrating Cribl Edge with your existing IT monitoring and alerting tools to gain comprehensive visibility into its health and performance.

Cribl Edge and Resource Constraints

Cribl Edge is designed to be a lightweight agent that often runs on machines that have limited resources. Unlike Cribl Stream, Edge uses a single Worker Process to limit resource usage and avoid affecting other functions of that machine.

While this is beneficial for resource-constrained environments, it's important to remember two things when implementing Edge:

- **Data consumption:** A single Worker Process can handle a limited amount data. If you try to ingest a massive amount of data, the process might become overloaded. For guidelines, see Performance Considerations and Sizing and Scaling.
- **Data manipulation:** Extensive manipulations within Pipelines can also strain the single Worker Process. Be strategic about the transformations you apply to the data.

Key Metrics to Monitor

Cribl Edge offers you multiple ways to monitor your deployments in order to get an operational view of the health of your Fleets, Edge Nodes, Sources and Destinations, and more.

Edge Node Status

The Edge Nodes page provides status information for each Edge Node in the selected Fleet. You can use this page to quickly answer the following questions:

- Is the agent running?

- Is it connected to Cribl Leader?

Edge Node Connectivity

To verify the connectivity of your Edge Nodes, check the **Health** column in the **Edge Nodes** page. A status of "alive" indicates that the Edge Node is running and successfully communicating with the Cribl Leader.

For details, see View Edge Node Status Information and Tracking Data Activity for an Edge Node.

Set Up Continuous Monitoring for Your Edge Nodes

While the Edge Nodes page provides essential information, it's important to note that disconnected Edge Nodes may be automatically removed after a certain period. To ensure continuous monitoring of your Edge Node health and performance, consider implementing centralized monitoring with a third-party solution like Grafana. By configuring your Edge Nodes to send internal metrics to a centralized monitoring solution, you can gain real-time insights into their statuses and troubleshoot potential issues proactively. For details and resources, see the Centralized Monitoring section below.

Sources and Destinations Health

To maintain optimal performance and data reliability, monitor the health of your Sources and Destinations at both the Edge Node and Fleet levels. Monitoring at the Edge Node level provides granular insights into the performance of individual Nodes, while Fleet-level monitoring offers a broader view of overall system health and potential bottlenecks.

You can monitor and track the health of Sources and Destinations associated with an Edge Node on the **Edge Nodes** page. By filtering the list to display only Edge Nodes with "Warning" or "Error" statuses for their Sources or Destinations, you can quickly identify and troubleshoot potential issues. For details, see Monitor Source and Destination Health on Edge Nodes and Filter Edge Nodes.

Alternatively, you can monitor the health of Sources and Destinations within a specific Fleet. To do this, navigate to the desired Fleet, select **Health**, and then choose either **Sources** or **Destinations**. This will display a color-coded health status for each Source or Destination, providing a quick visual overview of their operational status. For details, see Overall Source and Destination Health.

Data Throughput

There are several pages you can use to monitor the rate at which data is collected and transmitted (data throughput) across your Edge Nodes. This information can help you determine if an Edge Node is capable of handling the current data generation rate and if data is being successfully transmitted to Destinations.

- **Cribl Edge home page:** The Cribl Edge home page displays the aggregate data for all Fleets, Subfleets, and Edge Nodes. The charts display information about traffic in and out of the system. For details, see

Monitor Health on the Edge Home Page.

- **Fleets page:** The **Fleets** page gives you access to more information about each of your Fleets and the health of Sources and Destinations associated with the Fleets. You can view the **Monitor** tab of a selected Fleet to analyze the traffic flowing through your Fleet. For details, see Monitor Fleet Information.
- **Edge Nodes page:** To monitor the activity of your Edge Node and identify potential issues with Sources, Destinations, Routes, Pipelines, or Packs: Navigate to the **Edge Nodes** page, pick a specific Node and click anywhere on the row to launch the **Node Info** modal. Then, select **Data Activity**. This will provide detailed insights into the Node's data flow without teleporting into the Node itself. For details, see Viewing the Data Activity for a Node.

Resource Utilization

To effectively monitor the health and performance of your Edge Nodes, use node and process information as well as details from logs.

Sources of node and process information include:

- **System Activity** tab: Gain a detailed view of your Edge Node's health, including CPU usage, memory consumption, network traffic, and disk operations. Please note that these metrics represent the total system usage and may not solely reflect the resource utilization of the Edge Node itself. To view **System Activity**, click the row for an Edge Node. For details, see Viewing the System Activity for a Node.
- **Process-Level Monitoring**: Delve deeper into specific processes, such as `cribl` processes, to pinpoint resource bottlenecks and optimize performance. For details, see Processes.

Logs to analyze include:

- **API Server Logs**: Examine the `cribl.log` file at `$CRIBL_HOME/log/cribl.log` for high-level system telemetry and license validation information.
- **Worker/Edge Node Process Logs**: Analyze the `cribl.log` files at `$CRIBL_HOME/log/worker/N/cribl.log` to scrutinize detailed resource utilization metrics for each Worker/Edge Node Process at a granular one-minute interval.

Combine these approaches to proactively identify and address potential performance issues and ensure optimal data processing and system stability.

Effective Monitoring Strategies

This section explores various strategies for effectively monitoring your Cribl Edge deployments. We'll delve into Edge Node logging, centralized monitoring, and network monitoring techniques to ensure optimal performance and data flow.

Edge Node Logging

Cribl Edge logs at the default `Info` level, providing a good balance between detail and performance for most environments. Analyze logs regularly to identify trends and potential issues.

If you encounter a specific issue or need more detailed information, Cribl Edge settings allows you to adjust the log level for individual channels. However, frequently changing log levels from the default `Info` might indicate an underlying issue requiring further investigation.

By carefully selecting the appropriate log level, you can obtain valuable troubleshooting information without compromising performance. For details, see Logging Settings.

Centralized Monitoring

Centralized monitoring allows you to efficiently collect and analyze data from multiple Edge Nodes. This approach helps you proactively detect issues by enabling you to:

- **Implement alerts:** Set alerts for critical metrics, such as agent failures or performance degradation.
- **Create dashboards**: Visualize key metrics and gain valuable insights into the overall system health.

Combine Cribl Edge with your existing IT monitoring tools to gain a unified view of your deployment's health. This allows you to identify and address issues before they impact performance. By sending Cribl Edge logs to platforms like Elastic, Grafana, or Loki, you can:

- **Centralize log management:** Consolidate logs from various Sources for efficient analysis and troubleshooting.
- **Create advanced dashboards:** Visualize complex metrics and correlations to gain deeper insights.
- **Implement sophisticated alerts:** Set up alerts based on specific conditions and triggers to proactively address issues.

For details, see Monitoring your Infrastructure with Cribl Edge. Additionally, you can use CriblVision to troubleshoot specific issues in CriblSearch, Grafana, and Splunk.

Network Monitoring

Monitor network traffic between the Edge Node and the Cribl Leader to identify and resolve network-related issues. Use third-party network monitoring tools to track network performance and troubleshoot connectivity problems efficiently.

Troubleshooting Edge Nodes

To troubleshoot potential issues with your Edge Nodes, consider the following:

- Start by verifying the Edge Node's configuration, including connection settings, data collection intervals, and permissions. For details, see Configuring Cribl Edge.
- Analyze the logs in Cribl Edge. For details, see Search Internal Logs. To isolate issues related to a specific Edge Node, filter the logs by the Edge Node's IP address. This will help you identify any error messages or exceptions that may require attention.
- Check network connectivity between the Edge Node and the Cribl Leader by reviewing the Health column on the Edge Nodes page.
- To monitor Edge resource utilization, examine the API server logs and Worker Process logs. These logs provide detailed information on CPU usage, memory consumption, network traffic, and other relevant metrics. You can also monitor the System activity for the entire host by Viewing the System Activity for a Node.

Refer to Common Challenges on the Edge topic for other troubleshooting tips.

7.4.6. Migrate From Third-Party Agents To Cribl Edge

Migrate your data collection infrastructure from third-party agents to Cribl Edge

---

This guide provides strategic considerations for migrating your data collection infrastructure from traditional third-party agents to Cribl Edge. Whether you're migrating from Splunk Universal Forwarders (UFs), Fluentd, Filebeat, or other agent technologies, the fundamental migration principles remain consistent.

Any migration to Cribl Edge generally follows the same path:

1. Decide on your migration strategy
2. Complete pre-migration planning
3. Implement your Edge configuration
4. Rollout your changes in a controlled environment

We'll examine each phase and what you need to consider along the way.



This is a considerations guide, not step-by-step instructions. Complex migrations require careful planning and benefit from professional services engagement.

For specific details on UF migrations, see: Migrating from Splunk Universal Forwarders to Cribl Edge.

Migration Strategy Overview

Your migration success depends heavily on choosing the right approach from the start. The complexity of your current deployment, available resources, and timeline constraints will all influence which path you should take. Understanding these options upfront will save time and reduce risk throughout the migration process.

Choosing Your Migration Approach

Before beginning any migration, you must decide between two primary approaches:

- Customer-managed migration
- Professional services engagement

Customer-managed Migration

- **Best for:** Simple deployments with limited configuration complexity.

- **Challenges:** Trying to migrate complex agent deployments, beyond simple deployments, can be challenging and time-consuming. Requires deep understanding of both current infrastructure and Cribl Edge architecture.
- **Suitable for:** Organizations with dedicated resources and Cribl Edge expertise, or proof-of-concept implementations.

Professional Services Engagement

- **Best for:** Large-scale or complex agent deployments.
- **Provides:** Automated migration tools, in depth analysis, and enhanced support.
- **Recommended for:** Organizations requiring minimal downtime, deployments with extensive customization, time-sensitive migrations.

Pre-Migration Planning

This phase involves understanding your current environment, identifying potential challenges, and ensuring you have the necessary resources in place.

Environment Assessment

The first step in migration planning is gaining a complete understanding of your current deployment. This comprehensive assessment will inform every subsequent decision and help identify potential migration challenges before they become problems.

Begin by conducting a thorough inventory of your current deployment, document both essential infrastructure components and organizational factors that could impact your migration strategy.

Required inventory items:

- OS types and versions across your infrastructure.
- Current data volumes and peak load patterns.
- Existing input configurations and custom settings.

Additional considerations that may impact your migration:

- Company span, subsidiaries, separate entities.
- Network topology, connectivity, and bandwidth constraints.
- Change control processes for your infrastructure.
- Rollback plan documentation before beginning the migration.

Configuration Analysis

The goal of configuration analysis is to determine the data flow and topology of your current system, so that you can use this information to inform your Fleet design. For example, in environments using centralized configuration management (like Splunk's Deployment Server), the way your current server classes are organized will significantly impact your Edge Fleet design decisions.

Key areas to document include:

- Centralized configuration management systems and their hierarchies
- Application or configuration packages deployed to agents
- Client-to-configuration mappings
- Custom configurations and transformations

For example, in Splunk UF environments specifically, you would examine:

- Deployment Server configuration using serverclass.conf
- Splunk Apps stored in $SPLUNK_HOME/deployment-apps
- Client-to-serverclass mappings

Resource Requirements Planning

Inadequate resource planning is one of the most common causes of migration failures. Cribl Edge introduces new components and data flow patterns that require careful capacity planning. Understanding these requirements upfront prevents performance issues and data loss during the migration.

Leader Scaling

For Cribl.Cloud deployments, first assess your current Leader's performance before adding Edge Nodes. Evaluate whether your Leader is already overtaxed or operating smoothly, examining CPU utilization, memory usage, disk space availability, and disk I/O performance. For details, see Monitoring Health and Metrics.

Once you've confirmed your Leader can handle additional load, ensure Leader resources are scaled to support expected Fleet/Worker Group counts. To spin up the right number of connection processes, add the number of Edge Nodes you will deploy into the Edge Nodes Count field. This field crunches numbers in the background and automatically spins up the correct number of Edge connection processes for you. For on-prem deployments as well as other considerations, see Deployment Planning.

Cribl Stream Worker Nodes

Ensure Stream Worker Nodes receiving data from Edge Fleets are properly sized for peak load. Undersized infrastructure can cause performance bottlenecks and data loss during high-volume periods.

Data Routing and Flow

Cribl Edge offers more sophisticated data routing capabilities than traditional agents, but this flexibility requires careful planning to avoid performance bottlenecks and ensure reliable data delivery.

Key considerations:

- Configure appropriate outputs based on your data routing needs
- Consider routing complexity when routing from Cribl Edge to Cribl Stream then to Destinations
- Plan for backpressure management, especially when routing to multiple Destinations

Understanding Architectural Differences

Most third-party agent migrations involve fundamental architectural changes that affect every aspect of your migration planning. Understanding these differences is critical for designing an effective migration strategy.

Configuration Management Models

Traditional agents often use different configuration management approaches than Cribl Edge.

Composition vs. Inheritance

- **Traditional agents:** Many use composition models where a single agent can belong to multiple configuration groups or classes
- **Cribl Edge:** Uses inheritance where one Edge Node belongs to a single Fleet/Subfleet

For example, in Splunk UF environments, one UF can belong to multiple serverclasses, but in Cribl Edge, one Edge Node belongs to a single Fleet/Subfleet.

Fleet Design Strategy

Fleet design is perhaps the most critical decision in your migration. Poor Fleet architecture can lead to management complexity, performance issues, and operational challenges. The goal is to balance functionality with maintainability while leveraging Cribl Edge's strengths.

Fleet Rationalization Approach

- Review your current configuration management to understand what needs to be collected
- Use tags, Pipelines, and Cribl Stream to reduce configuration variability
- Consolidate similar configurations to minimize Fleet count
- Group by function rather than direct configuration translation

Fleet Count Optimization

- Keep Fleet counts to a minimum for optimal performance
- Avoid direct translation of existing configuration groups to Edge Fleets (leads to Fleet proliferation)



Direct translation of existing configuration management structures to Edge Fleets often leads to management complexity and potential performance issues.

- Consider creating Fleets for unique combinations of configuration requirements only when necessary

Configuration Translation Challenges

Understanding these caveats upfront allows you to plan alternative approaches and avoid migration roadblocks.

Automatic Metadata Population

One advantage of Cribl Edge is its automatic handling of common metadata fields, reducing configuration complexity compared to traditional agents. Cribl Edge automatically populates these metadata fields without additional configuration:

- `sourcetype`
- `index`
- `source`
- `host`
- `connection_host`

Configurations Requiring Alternative Approaches

Some agent configurations have no direct equivalent in Cribl Edge and require additional steps. Common examples include:

- **Path-based host extraction:** Some agents (like Splunk UF) can automatically extract the hostname from the file path itself. For example, the `host_regex` and `host_segment` settings from UF don't translate directly. However, you can achieve the same functionality using Cribl Edge's Pipeline processing to extract and transform the hostname data as needed.
- **File handling behaviors:** Some agents often have granular control over how they interact with files. Cribl Edge may not have equivalent settings for every file handling behavior. For example, symbolic linking or file opening behavior.
- **Custom transformations:** Some agents can perform data transformations at the collection point. Cribl Edge operates on a different model where most data transformation happens in Cribl Stream rather than at the edge. While Cribl Edge can do some processing, it's not designed to replicate all the transformation capabilities that agents might have built-in.

Alternative Implementation Methods

- Create corresponding Pipelines in Cribl Edge
- Use Edge Node metadata mapping for system information
- Implement data transformation through Routes (all inputs forward to Routes)

Migration Implementation Considerations

Once you've completed your planning phase, the implementation phase requires attention to configuration details and platform-specific considerations.

Configuration Review Areas

Each configuration area requires careful review to ensure optimal performance and functionality.

File Monitor Settings

- Determine whether to start from the beginning or end of files
- Consider the impact of large file sizes or many old events
- Assess network bandwidth and Destination capacity implications

System State Configuration

The System State Source is enabled by default on Cribl Edge Fleets, but you can disable the Source or specific collectors if System State data is not needed.

Pipeline Configuration

- Review existing agent configurations and enable similar functionality in Cribl
- Optimize regexes for performance
- Consider data transformation requirements previously handled in agents

Persistent Queue Management

- Review Persistent Queue settings in your agents, Cribl Edge, and Stream
- Ensure proper queue sizing for expected data volumes

Platform-Specific Considerations - Windows Environments

Windows environments present unique challenges during migration, particularly around policy management.

Group Policy Objects

Check for Group Policy Objects (GPOs) that might automatically reinstall or enable removed third-party agents, and update policies before migration to prevent conflicts.

PowerShell Usage

For optimal performance on Windows Edge Nodes, ensure the “Use Windows Tools” setting is disabled for all data Sources. When disabled, the system uses native Windows Management Instrumentation (WMI) API methods for faster and more reliable data collection. When enabled, the system falls back to legacy PowerShell cmdlets.

The PowerShell-based collection methods will be removed in future releases as native methods provide superior speed and reliability.

Policy Management

Some sites might have policies to re-enable/re-install agents if missing/removed. Update or exclude policies before migration.

Windows Event Log Read Mode

When configuring the **Windows Event Log** Source, always review the **Read mode** setting. The default is **From last entry**, which only collects new events. While this is the recommended setting, you have the option to

change it to read the **Entire log**. For machines that have been in use for a long time, this can cause a massive, one-time spike in data ingestion as it sends years of historical events. This sudden influx can easily overwhelm your infrastructure, causing performance issues and data loss. Sticking with the default setting helps you avoid this risk.

Addressing Inheritance Limitations

When working with Edge's inheritance model, be aware of these challenges:

- **Configuration integrity**: Changes to a Source/Destination in a Subfleet can break inheritance from the parent
- **Limited flexibility**: Sources that allow only one configuration at a time create inheritance challenges when customization is needed

Migration Rollout Strategy

The rollout phase is when your plans become reality. A good rollout ensures things go smoothly, problems are fixed fast, and your data collection isn't interrupted.

Controlled Rollout Process

A controlled rollout allows you to validate your migration approach on a small scale before committing to full deployment. This approach significantly reduces the risk of widespread issues and provides opportunities to refine your process.

Progressive Batch Sizing

- Start with test servers
- Then proceed to small batches (e.g., 200 machines)
- Progress to larger batches (400, 800, 2000 machines, etc.)
- Validate system performance and data flow at each stage
- Only proceed when previous stage is validated

Validation Requirements

- Execute controlled rollout with proper validation at each stage
- Address any issues early in rollout before expanding
- Maintain visibility into migration rollout progress

Continuous Monitoring

To ensure a smooth migration, you need to monitor everything. This means watching both how things are performing technically and making sure your important data is flowing correctly. Additionally, monitor the Edge Nodes themselves as they get onboarded to track their connectivity status. Cribl Edge v.4.13 introduced Disconnected Node Tracking to help you identify and troubleshoot nodes that lose connection during the migration process.

Resource Utilization

Monitor Cribl Edge, Leader, and Stream components continuously to watch for performance bottlenecks or data loss indicators. Validate data flow and collection accuracy throughout the process.

Rollback Capabilities

Maintain (previously-documented) rollback capabilities in case issues are discovered.

TL;DR

Migrating from third-party agents to Cribl Edge requires careful planning and a systematic approach. The key to success lies in understanding the architectural differences between your current system and Cribl Edge, properly planning your Fleet design, and executing a controlled rollout process.

Remember that this is a considerations guide, and complex migrations often benefit from professional services engagement to ensure success and minimize risk to your data collection infrastructure.

7.5. Deployment Planning

Plan your Cribl Edge deployment, taking into account requirements, deployment types, scaling, and billing.

When you're planning an Edge deployment, consider these key factors:

- The amount of data (metrics, logs, custom command outputs, and so on) that you plan for Edge to collect from the endpoint and ingest per unit of time. For example, MB/s or GB/day.
- The amount of processing that will happen on incoming data. For example, are there a lot of transformations, regex extractions, parsing functions, field obfuscations, and so on?
- Routing and/or cloning: Is most data going to a single Destination, or is it being cloned and routed to multiple places? This is important, because Destination-specific serialization tends to be relatively expensive.
- Deploying onto servers with no internet access: If you plan to deploy Edge Nodes on air-gapped on-prem servers (servers with no internet access), you must ensure that every Edge Node can communicate with a Leader that is also on-prem (that is, not in Cribl.Cloud).



See Requirements for detailed information about system and OS requirements for a Cribl Edge deployment.

Types of Deployment

- Cribl.Cloud: Quickly launch a Cribl-managed deployment of the combined Cribl applications suite (Edge, Stream, Search, and Lake). In Cribl.Cloud, Cribl provisions and manages all infrastructure on your behalf.
- **Managed Edge**: A central Leader Node controls, manages, and pushes configuration updates to all managed Edge Nodes. This deployment type provides a centralized view for multiple Edge instances, which is beneficial for large-scale deployments of Cribl Edge.
- **Single-instance**: Single-instance Edge Nodes operate without a central Leader. You will configure and manage Edge Nodes locally. Single-instance Edge is suitable for smaller deployments or environments with limited network connectivity.

When you have a large number of Edge Nodes to manage, you can use Fleets to group Nodes together and apply configurations centrally. Use Fleet Management to accommodate increased load. See Add and Update Edge Nodes to streamline deploying Edge Nodes via scripting.

Leader Scalability

Our System Requirements suffice for Leaders handling up to 10,000 Edge Nodes. To support more than 10,000 Nodes, see our Planning Large-Scale Deployments of Cribl Edge guide.

For large on-prem deployments handling high traffic, review the recommended OS-level tuning detailed in OS Tuning for Large Deployments.

About Billing and Usage

Cribl.Cloud offers a one-page view of your total credits consumed and remaining credits for the current active contract in place. This is called Cribl FinOps Center. Here, you can view per-product consumption for the Cribl products you use. The **Invoices** tab lets you download and export your usage by month. For more details, go to the FinOps Center docs.

On-Prem Licenses

If you're deploying on-prem, there are three license options. The Cribl Plans page provides details feature comparisons for each license.

You can configure Connected Environments to use the Cribl.Cloud FinOps Center billing model with your on-prem deployments instead of managing multiple licenses.

Permissions and Rights

You do **not** need elevated privileges to install Edge. However, you will need:

- Administrator (or equivalent) rights in order to enable Edge to start on boot.
- Sufficient rights to access the resources that need monitoring - files, metrics, and so on.

Network Requirement (DPI/IPS Exclusion)

Deep Packet Inspection (DPI) and Intrusion Prevention Systems (IPS) on network devices situated between your Edge Fleets can cause high resource utilization and connection instability in all environments (both Cribl.Cloud and on-prem). Explicitly exclude all Cribl data streams from DPI/IPS inspection for stable operations and optimal performance.

Performance Considerations

As with most data-collection and -processing applications, Cribl Edge's expected resource utilization will be proportional to the data volume and type of processing. For instance, a Function that adds a static field on an event will perform faster than one that applies a regex to find and replace a string.

- Processing performance is proportional to CPU clock speed.
- All processing happens in-memory.
- Processing does not require significant disk allocation.

Thinking about your planned Edge deployment as a whole, it's critical to consider **cardinality**: how many Fleets, of what size, will send metrics to a given Leader. In a very low-cardinality deployment, a Leader Node might aggregate metrics from one small Fleet; in a very high-cardinality deployment, a Leader Node might aggregate metrics from many large Fleets.

Too many Edge Nodes sending too many metrics can degrade Leader performance and/or integrity. By carefully choosing what metrics each Fleet sends to its Leader Node, you can ensure that you collect the metrics most important to you, within the limits of the Leader's capacity to process them.

As part of your deployment planning, decide which of the following four sets of metrics each Fleet in your deployment will send to its Leader Node:

- **Minimal**: Limited to metrics for events and bytes in and out. Recommended for high-cardinality deployments.
- **Basic** (the default set): All the metrics required to display all monitoring data available in the Edge UI. Contains all the metrics in the **Minimal** set, and more.
- **All**: All metrics, sent without filtering.
- **Custom**: Only metrics that match a JavaScript expression that you define.

For a listing of the metrics each set provides, see Control Metrics in Cribl Edge.

Configure Edge Node Count

Enter the estimated number of Edge Nodes in this deployment, which is used to calculate how many Edge-specific connections service processes to spawn. Changing this number could restart any currently running service processes. If left blank, defaults to 0.

To configure the Edge Node Count:

1. From **Settings**, select the **Global** tab, then select **General Settings**.
2. In an on-prem deployment, select **Limits**, and then **Others**.
3. Enter the number of Edge Nodes in the **Edge Nodes count** field.

Configure Connection Processes

To configure the connection processes for your Edge Nodes in an on-prem deployment:

1. From **Settings**, select the **Global** tab, select **Service Processes**, and then select **Services**.
2. Confirm that **Connections listener number of processes** is set to its default `1`.



The **Connections listener number of processes** is automatically determined by the **Edge Node Count** setting you enter above. Any value you enter manually will be overridden by this calculation.

7.5.1. OS And System Requirements

For a successful Cribl Edge deployment, verify that your system meets the minimum hardware and software specifications outlined below.



All the quantities listed below are **minimum** requirements. Adjust them as needed based on your specific use case.

OS Requirements

You can install Cribl Edge Nodes on Linux, Windows, and macOS systems.

Linux Requirements

Cribl Edge Nodes can run on any common Linux distribution that fulfils the following criteria:

- 64-bit kernel >= 3.10
- glibc >= 2.17
- OpenSSL >= 3.0 (if FIPS mode is enabled)

Highly customized or rare Linux distributions might not be supported even when based on a supported kernel version.

Enforcing mode in SELinux is supported, but not required.

Tested Linux Platforms

Cribl tests the following Linux versions to confirm compatibility with the supported kernel. This list does not constitute a comprehensive record of all supported distributions and versions.

- Ubuntu 24.04
- RHEL 9, 9.3
- CentOS Stream 9
- SUSE Linux Enterprise Server 15.6
- Amazon Linux 2023

As of October 16, 2024, Cribl has deprecated support for the CentOS Linux 7 distribution, which reached its end of life on June 30, 2024.

Windows Requirements

Cribl Edge supports these versions of Windows:

Version	Notes
Windows 10 and 11	You can deploy Cribl Edge on laptops, desktops, and workstations running Windows 10 and 11. Keep these things in mind when deploying Cribl Edge on Windows 10 and 11 workstations: • Cribl Edge might lose some data sent to TCP Destinations after a workstation resumes from a suspended or sleep state. This happens when the data is in local network buffers but hasn't been sent out to the Destination. Instead, you can send data to an HTTP Destination. • You might see partially broken events after a laptop resumes from a suspended state. • HTTP Destinations might contain duplicate events. • Some users have observed excessive logging when a Leader is unavailable to receive heartbeat messages from an Edge Node.
Windows Server 2016 Windows Server 2019 Windows Server 2022 Windows Server 2025	Functionality is the same as generally supported Windows Server platforms.

MacOS Requirements

Cribl Edge supports the following macOS versions: 26 (Tahoe), 15 (Sequoia), 14 (Sonoma).

You can install Cribl Edge on both Intel and Apple silicon processors.

System Requirements

Edge Nodes should have sufficient CPU, RAM, network, and storage capacity to handle your specific workload. While minimum specifications are provided, optimal configurations vary widely. Consider data volume, types, processing needs, and organizational standards when selecting hardware.

Make sure to test the performance of your Edge Nodes before deploying to production.

Minimum application requirements for Edge Nodes are:

- ~1 Ghz processor
- 1024 MB RAM
- 5 GB free disk space per volume for operational purposes

If your installation uses multiple volumes (for example, one volume to store persistent queues and one for Destination staging directories), 5GB is the minimum disk space for each of those volumes. You can configure the disk space limit per Fleet with the **Min free disk space** setting.



See also Cribl Stream Requirements to learn about the requirements for the Leader Node.

Browser Support

The Cribl Edge UI can run on Chrome, Firefox, Safari, and Microsoft Edge, on their five most-recent versions.

7.5.2. Plan Large-Scale Deployments Of Cribl Edge

Learn how to scale Cribl Edge deployments to support one Leader managing up to 250,000 Nodes.

Our standard Deployment Planning guide is a great place to start if you intend to deploy 10,000 or fewer Nodes in your Cribl Edge environment.

For larger deployments, a Leader can support a maximum of 250,000 Edge Nodes when metrics are set to **Minimal** mode and 100,000 Edge Nodes with metrics set to **Basic**.

This guide covers:

- The specific Leader configuration we recommend for best performance.
- Overview of upgrading a large Cribl Edge deployment.
- Limitations to the UI resulting from a large number of Edge Nodes.

Recommended Leader Configuration

The base configuration detailed on Cribl Stream Sizing and Scaling is a good starting point.

From there, you can configure these settings on the Leader so it can manage up to 250,000 Edge Nodes:

- Add 2GB of RAM for every 10,000 Edge Nodes.
- Deploy one vCPU core for every 10,000 Edge Nodes.
- For Cribl Edge to support up to 100,000 Nodes, you can configure internal metrics to **Basic** in **Fleet Settings**.
- For 250,000 Nodes, you must configure internal metrics to **Minimal** in **Fleet Settings**.



You should configure these metrics for every Fleet if your Edge Node count is more than 10,000. See Performance Considerations for more information.

- To spin up the right number of connection processes, add the number of Edge Nodes you will deploy into the **Edge Nodes Count** field. This field crunches numbers in the background and automatically spins up the correct number of Edge connection processes for you.
- There is no change to the default values when pushing config bundles from the Leader.

Recommended Fleet Configuration

There are additional settings you'll need to configure for each Fleet you intend to create:

- Add 1 vCPU per Fleet.
- Add 2GB of RAM per Fleet.

Upgrading a Large Number of Edge Nodes

The upgrading framework allows Edge Nodes to pull down upgrade packages from the Leader asynchronously, rather than the Leader pushing packages to Nodes all at one time. Nodes communicate to the Leader every 60 seconds via a heartbeat.

Additionally, Nodes request upgrade packages from the Leader in batches of 500. This allows you to upgrade Nodes without overloading the Leader, even with a large number of Nodes.



In our controlled test environment, we found that upgrading can take 25-40 minutes. This is an expected time frame to upgrade that number of Nodes.

User Interface Limitations

For Fleets with a large number of Edge Nodes, the UI is limited in certain places where a complete list of Nodes can't be displayed.

Searching for Nodes in the Explore Tab

The Node field in the **Explore** tab returns only the first 100 matching Nodes. When you start typing, the results match the search query and are sorted alphabetically by hostname.

Previewing Node Mappings

Preview Node mappings by selecting **Mappings** in the Leader Node's sidebar. (Only the first 10,000 mappings are displayed.)

Other Limitations

- **Logs**: The drop-down where you can select an Edge Node will display a maximum of 100 Nodes.
- **Import Edge Data**: The **Import Edge Data** window in Cribl Stream returns a maximum of 100 Edge Nodes.
- **Test tab**: The **Test** tab on Destinations displays a maximum of 100 results.
- **KMS Status report**: In **Fleet Settings**, the **KMS Status report** (under **Security**) does not display all Nodes. Instead, search for your desired Node.

7.6. Set Up Leader And Edge Nodes

Configure a Leader Node and a Cribl Edge Node.

This page covers how to:

- Configure a Leader Node
- Configure an Edge Node

Configure a Leader Node

You can configure a Leader Node either through the UI or through the `instance.yml` config file.

Use the UI

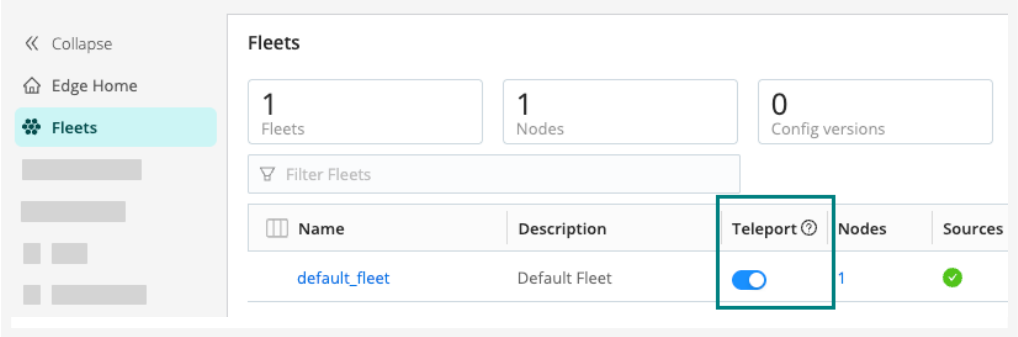
1. Navigate to **Settings** > **Global** > **Distributed Settings** > **General Settings**.
2. Select **Mode**: **Leader**.
3. Select the **Leader Settings** menu.
4. Confirm or enter the required Leader settings (**Address** and **Port**).
5. Confirm Leader API Access: Toggle **Forward to Leader API** on (default) to enable API requests on the distributed port and provide backward compatibility.
6. Customize the optional settings if desired.
7. Select **Save** to restart the Cribl Edge server.

Edge Node UI Access

This useful option enables you to click through from the Leader's **Edge Nodes** page to an authenticated view of each Node's UI. We call this Teleporting into an Edge Node. The instructions below correspond to enabling the `groups.yml` file's `workerRemoteAccess` configuration key.

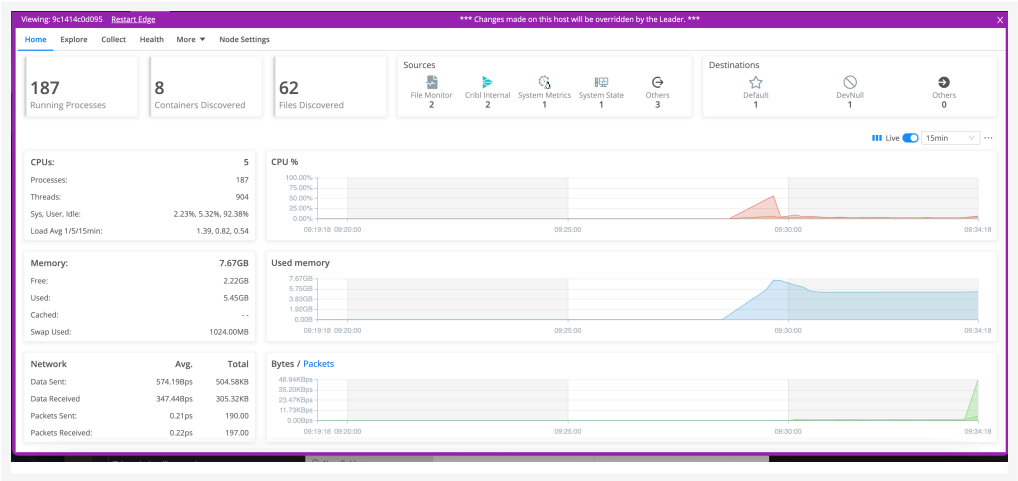
To enable teleporting (remote UI access) to Nodes from the Leader's UI:

1. Select **Fleets** from the sidebar.
2. For each desired Fleet, toggle on **Teleport**.



Manage Fleets

3. From the sidebar, select **Edge Nodes**.
4. On the **Edge Nodes** page, click the link for any Node you want to teleport into.
   To confirm that you are remotely viewing a Edge Node's UI, Cribl Edge displays a purple border, with a badge labeled **Viewing host:** `<host/GUID>`.



Authenticated view of an Edge Node



The Leader will override any changes that you make directly to this Edge Node.

Use the YAML Config File

In `$CRIBL_HOME/local/_system/instance.yml` (`C:\ProgramData\Cribl\local\_system\instance.yml` on Windows), under the `distributed` section, set `mode` to `master`:

$CRIBL_HOME/local/_system/instance.yml or C:\ProgramData\Cribl\local_system\instance.yml

```
distributed:
  mode: master
  master:
    host: <IP or 0.0.0.0>
    port: 4200
    forwardToLeaderApi: true
    tls:
      disabled: true
    ipWhitelistRegex: /.&ast;/
    authToken: <auth token>
    enabledWorkerRemoteAccess: false
    compression: none
    connectionTimeout: 5000
    writeTimeout: 10000
```

Copy

Persisting Socket Connections

A distributed deployment creates socket files for inter-process communication (IPC) between the Leader and distributed processes and services. These sockets are essential for ensuring that Edge Nodes successfully connect to the Leader, and for certain metrics services. On the Leader's host, the default location for these files is the operating system's temp directory (for example, `/tmp`).

Many Linux distros maintain a system cleaner service (for example, systemd-tmpfiles) that removes files from this directory periodically, such as every 10 days. If Cribl's sockets are removed, this breaks certain UI pages, such as those for Fleets and Monitoring. You can protect the sockets in either of two ways.

Block the Cleaner

Stop the host OS from cleaning socket files out of `/tmp/cribl-*` subdirectories. For example, on Amazon Linux 2 instances, add a new `tmp.conf` file to `/etc/tmpfiles.d` with the line: `X /tmp/cribl-*`.

To restart the system cleaner here and reload its configuration, use this command:

```
sudo systemctl restart systemd-tmpfiles-clean.service
```

Copy

Move the Sockets

Alternatively, you can move Cribl's socket files to a different directory. This directory must be outside your operating system's temp directory, and the user that owns the `cribl` process must have the necessary permissions to write to it.

In Unix-like operating systems, the maximum length for Unix domain socket paths is 108 bytes. To minimize the path length, use a directory that is close to the root, such as `/var/tmp`.

In the UI, you specify the directory at **Settings** > **Global Settings** > **System** > **Distributed Settings** > **Leader Settings** > **Helper processes socket dir**.

Configure an Edge Node

On each endpoint, you can configure Cribl Edge variously through the UI, the `instance.yml` config file, environment variables, or the command line.

Use the UI

1. Navigate to **Settings** > **Global** > **Distributed Settings** > **General Settings**.
2. Select **Mode**: **Managed Edge**.
3. Select the **Leader Settings** menu.
4. Confirm or enter the required **Address** (for example, `criblleader.mycompany.com`). Customize the optional settings if desired.
5. Select **Save** to restart the Cribl server.

Use the YAML Config File

In `$CRIBL_HOME/local/_system/instance.yml` (`C:\ProgramData\Cribl\local\_system\instance.yml` on Windows), under the `distributed` section, set `mode` to `managed-edge`:

```
$CRIBL_HOME/local/_system/instance.yml or C:\ProgramData\Cribl\local\_system\instance.yml
```

```
distributed:
  mode: managed-edge
  envRegex: /^CRIBL_/
  master:
    host: <master address>
    port: 4200
    authToken: <token here>
    compression: none
    tls:
      disabled: true
    connectionTimeout: 5000
    writeTimeout: 10000
  tags:
       - tag1
       - tag2
       - tag42
  group: teamsters
```

Copy

Use Environment Variables

You can configure Edge Nodes via environment variables, as in this example:

```
CRIBL_DIST_LEADER_URL=tcp://${CRIBL_DIST_TOKEN:-criblmaster}@masterHostname:4203
./cribl start
```

For additional details, see Configure Environment Variables.

Use the Command Line

You can configure an Edge Node using CLI commands of this form:

```
./cribl mode-managed-edge -H <master-hostname-or-IP> -p <port> [options] [args]
```

7.7. Connect Nodes To Leader Through Cribl Outpost

Simplify Node-to-Leader communication with Cribl Outpost.

Cribl Outpost simplifies management of deployments by acting as a relay between individual Edge Nodes or Stream Workers and the Leader Node.

Once you establish an Outpost and configure it to accept traffic from Edge Nodes or Stream Workers, control plane communication between those Nodes and the Leader will pass through the Outpost.

Cribl Outpost is ideal in situations where you can't establish direct communication between Edge Nodes or Workers and the Leader, for example due to restrictions in opening up firewall ports.

Using an Outpost limits the need to set up third-party proxies and simplifies maintenance by reducing the complexity of firewall setups.

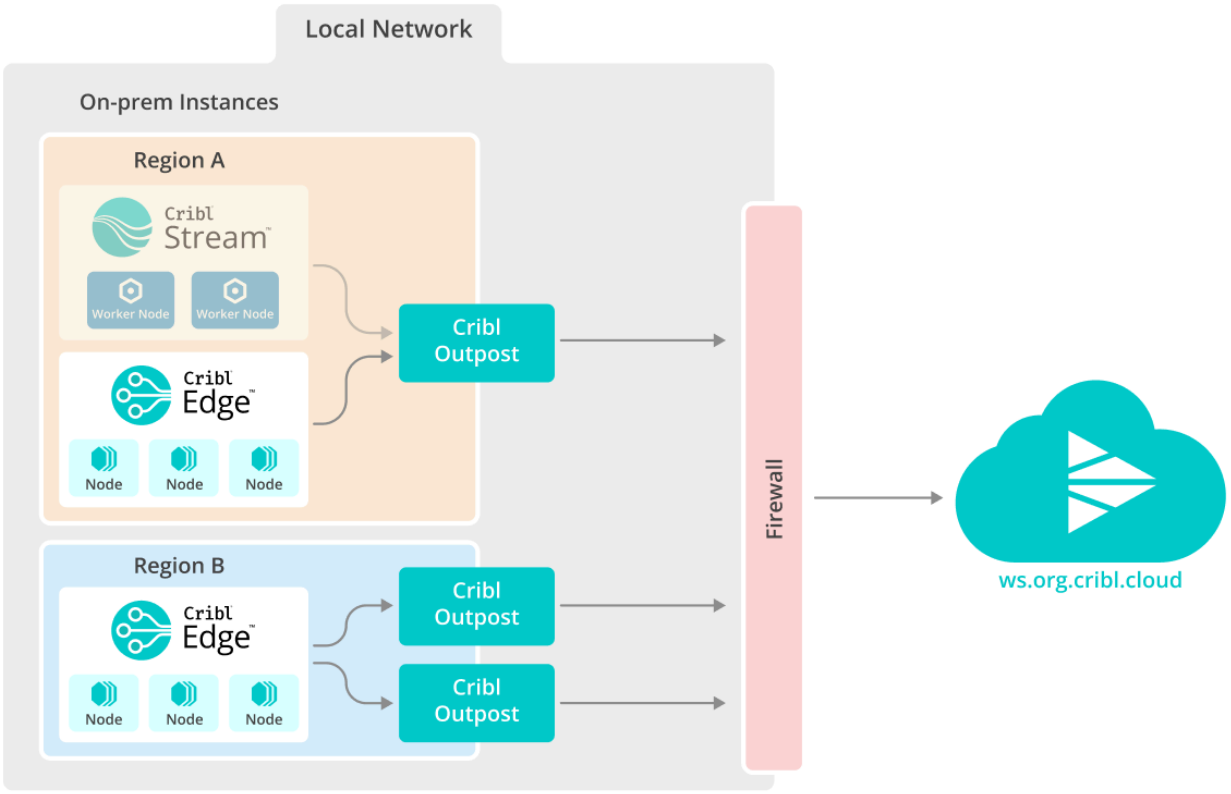


Cribl Outpost requires an Enterprise plan or Enterprise license. For details, see Pricing.

If you connect an Outpost to a Leader that does not have a proper plan or license, the Outpost node will close the listener for communication incoming from Edge Nodes or Workers, but will be shown as healthy on the Leader.

How Outpost Works

An Outpost is a special type of Node that rests between Edge Nodes or Stream Workers and the Leader.



Overview of how Outpost connects Leader with Edge Nodes and Stream Workers

For an individual Edge Node or Worker, an Outpost is indistinguishable from the Leader: Nodes communicate with the Outpost the same way they would with a Leader, using the same configuration (for example, using the URL defined in the `CRIBL_DIST_LEADER_URL` environment variable or in `instance.yml`).

Instead of connecting directly to multiple Nodes (possibly thousands), the Leader maintains those connections with one Outpost only. This means you only need to configure communication lines such as firewall exceptions once per Outpost.

Outpost Capacity

You can connect a maximum of 28,000 Edge Nodes to a single Outpost. The specific number of possible connections up to this limit depends on the Outpost Node specifications, in particular on CPU and RAM.

The recommended specifications for serving a given number of Edge Nodes are as follows:

Number of Edge Nodes	vCPU	RAM	Connection listener processes
up to 10,000	2	4 GiB	1

7.7.1. Set Up Cribl Outpost

Create a Cribl Outpost to relay communication between Edge Nodes and the Leader.

Prerequisites

Cribl Outpost requires an Enterprise plan or Enterprise license. For details, see Pricing.

You can install Cribl Outpost on Linux only. Do not run an Outpost Node on the same host as a Stream Worker to avoid both Nodes competing for resources.

Create an Outpost

There are two ways of adding a new Outpost Node:

- from a binary file
- using the bootstrap script provided in the UI

Add an Outpost from a Binary File

To add a new Outpost Node using a downloaded binary file:

1. Go to the Cribl Download page and set the **Software** drop-down to `Cribl Suite (Edge and Stream)` for your processor architecture.
2. Select **Download now** to get the `.tgz` archive.
3. Un-tar the install package in your directory of choice:

```
cd /opt/
tar xvzf cribl-<version>-<build>-<arch>.tgz
```

Copy

4. Set the installation mode to `mode-outpost` to connect the Outpost to the designated Leader and allow Stream Workers to connect to it in turn:

```
/opt/cribl/bin/cribl mode-outpost /
   -H <leader-hostname-or-IP> /
   -p <port> /
   -u <token> /
   -S 1
```

Copy

`<token>` must match the Leader token, to ensure a connection with the Leader.
In Cribl.Cloud, get the token for your Workspace from the Edge Node modal (**Add/Update Edge Node**) from the **Auth token** field.
In an on-prem deployment, get the token from **Settings** > **Global** > **System** > **Distributed Settings** > **Leader Settings** > **Auth token**.

5. Configure the installation as a service and start it:

```
sudo /opt/cribl/bin/cribl boot-start enable -m systemd -u cribl
sudo systemctl restart cribl-outpost.service
```

Copy

At this point your Cribl Outpost is ready for use and you can start configuring Edge Nodes to connect to it.

Add an Outpost with a Bootstrap Script

To add a new Outpost Node using a CLI bootstrap cript:

1. From the sidebar, select **Outpost**, then select an Outpost Group.
2. Select **Add/Update Outpost Node**, then select **Add**.
3. Fill and modify the deployment option fields:
   - The **Install package location** defaults to `Cribl CDN`. If desired, change this to `Download URL`.
   - As needed, correct the target **Outpost Group**, as well as the **Leader hostname/IP** (URI).
   - As needed, correct the **User** and **User Group** to run Cribl as.
   - As needed, correct the **Installation directory**.
4. Optionally, add **Tags** that you can use for filtering and grouping Outposts. Use a tab or hard return between (arbitrary) tag names.
5. Copy the resulting script to your clipboard.

Finally, to add the Outpost, paste the script onto the command line where you want to set up the node, and execute it.

If your script successfully added the Outpost, you will see it in the **Outposts** list for your selected group.

Connect Nodes to an Outpost

Edge Nodes see Outpost in the same way as a Leader Node. This means that, to connect a Node to an Outpost, you only need to provide the Outpost listener URL that you defined with the `-O` parameter, and the Leader auth token.

The Outpost doesn't have a separate auth token. When configuring Node connection, use the auth token of the Leader.

Configure the connection either from Settings (**Global** > **Distributed Settings** > **Leader Settings**), or by setting the `CRIBL_DIST_LEADER_URL` environment variable:

```
CRIBL_DIST_LEADER_URL=tcp://<leader-token>@<outpost-hostname>:<outpost-port>
```

Copy

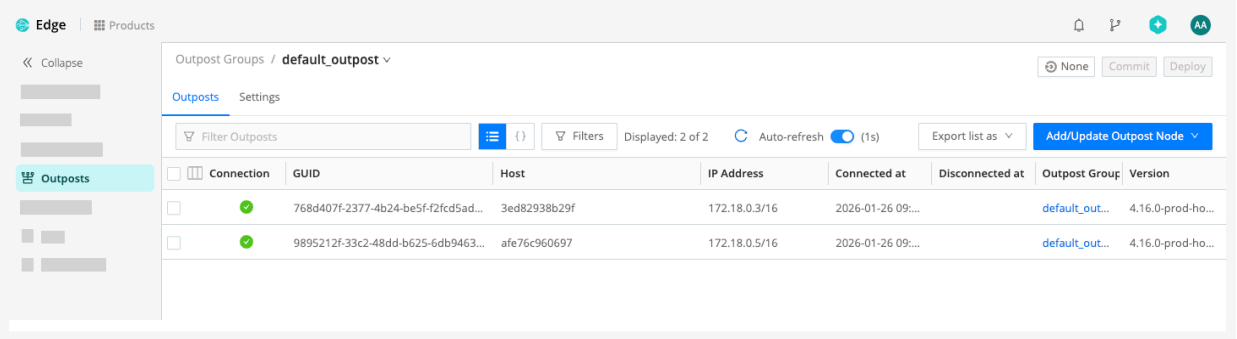
7.7.2. Manage Cribl Outpost

Control, diagnose, and upgrade your Cribl Outposts.

View Outposts

The **Outposts** page displays all Outposts configured in your deployment, including those that are currently disconnected.

1. On the top bar, select **Products**, and then select **Edge**.
2. In the sidebar, select **Outposts**. You will see a list of all Outpost Groups.
3. Select any group to see its Outposts, including disconnected ones.



List of Outposts

4. Select any row in the table to see detailed information about the Outpost.

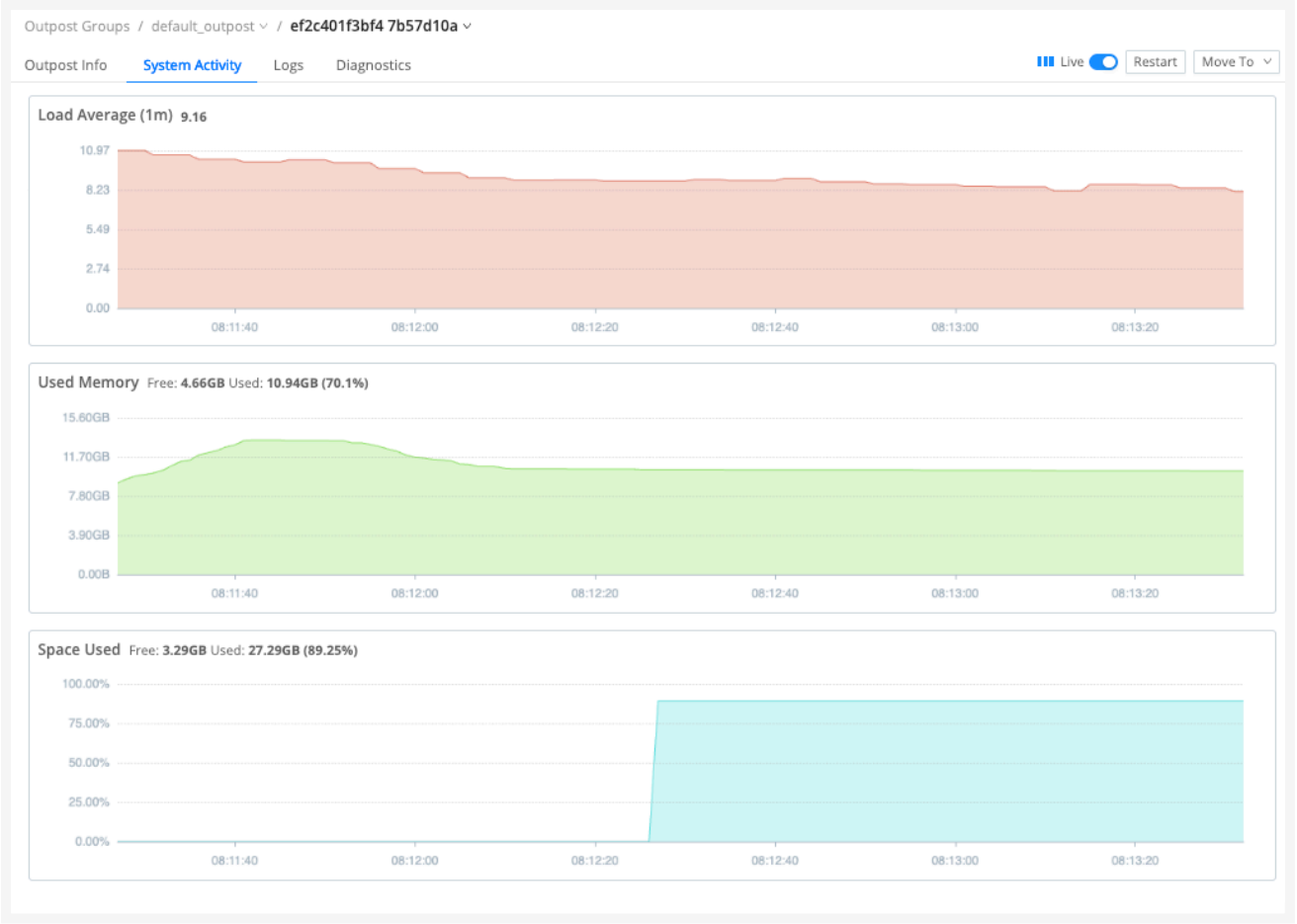
Alternatively, select **List View** above the Outpost Group table to view all Outposts in all groups.

To export the filtered list of Outposts in JSON or CSV format, select **Export list as** above the table.

Check Outpost Activity

To view the data flow through an Outpost Node, select an Outpost Group and an individual Outpost, then select the **System Activity** tab. This tab displays charts for:

- Load Average (1m)
- Amount of memory used
- Amount of disk space used



Outpost System Activity charts

View Outpost Logs

To view Outpost logs, consult the **Logs** tab in the Outpost screen. For more information about Outpost logs, see Internal Logs.

Create Outpost Diagnostic Bundle

In the outpost screen you also create a diagnostic bundle to share with Cribl Support as help in diagnosing any issues. To do so, on the **Diagnostic** tab, select **Create Diagnostic Bundle**. To learn more about diagnostic bundles, see Diagnose Issues.

You can also create diagnostic bundles by using the `diag` command.

Restart an Outpost

Required Permission: **Admin/Owner**

To restart an Outpost:

1. On the top bar, select **Products**, and then select **Edge**.
2. In the sidebar, select **Outposts**, then select one of the Outpost Groups.
3. Select the check box in the row of the Outpost or Outposts you want to restart and select **Restart Selected**.
4. Confirm with **Restart**.

Manage Outpost Groups

Outpost Groups let you manage configurations for Outposts in bulk. The **Settings** tab in Outpost Group view lets you configure preferences for the group. The settings available are a subset of the settings for a Worker Group (in Cribl Stream) or an Edge Fleet (in Cribl Edge).



Outposts created in versions older than 4.16.0 will always stay in the `default_outpost` group. To move those Outposts between groups, first upgrade them to version 4.16.0 or newer.

Outpost Connectivity

You can configure how long the Leader should keep track of disconnected Outposts by configuring the **Time to keep disconnected Outposts** Outpost Group setting. By default the Leader retains information about disconnected Outposts for one day. You can set it to any value between 0 (which means Outposts are removed within a few minutes) and 90 days.

Outpost Upgrades

Defining a target version for an Outpost Group lets you upgrade all Outposts from that group in bulk. See Upgrade Outpost for more information.

Move Outpost to Another Group

You can move an Outpost to another group by selecting its check box and selecting **Move To** below the Outpost table.

Moving an Outpost to another group can temporarily disconnect it from the Leader. If the new group has a target software version selected, the Outpost will upgrade to that version on move.

Moving an Outpost via the **Move To** button will be impossible for Outposts that have any `DIST_` environment variables set.

7.8. Manage Edge Node Config Bundles

Learn about Edge Node config bundles and how to manage them

Config Bundles

Config bundles are compressed archives of all config files and associated data that an Edge Node needs to operate. Both Leader and Worker Nodes automatically manage config bundles in the following ways.

Bundle Storage and Delivery

The delivery mechanism for config bundles depends on where your Leader resides:

- **Cribl.Cloud Leader**: Bundle storage is managed for you in a Cribl-owned Amazon S3 bucket. You do not need to configure S3 settings. Edge Nodes will automatically attempt to pull from the Cribl S3 bucket first, then fall back to the Leader over port `4200`. For details on preventing deployment delays by white-listing the required S3 and Leader endpoints, see Network Access for Config Bundles.
- **Customer-managed Leader**: By default, the Leader serves bundles directly. To reduce Leader load, you can store bundles in S3 using your own bucket. For details, see Store Bundles Remotely.

For details on ports, see Ports and Required Ports in Cribl.Cloud.

Bundle Lifecycle and Retention

Regardless of where bundles are stored, Cribl Edge maintains a cleanup policy to optimize disk space.

Leader Node Bundles

The Leader creates bundles upon Deploy, and manages them as follows:

- Bundles are wiped clean on startup.
- While running, at most 5 bundles per group are kept.
- Bundle cleanup is invoked when a new bundle is created.

Edge Node Bundles

The Edge Node pulls bundles from the Leader and manages them as follows:

- Last 5 bundles and backup files are kept.
- At any point in time, all files created in the last 10 minutes are kept.
- Bundle cleanup is invoked after a reconfigure.

Store Bundles Remotely

For customer-managed Leaders, you can optionally reduce Leader load by storing bundles in an Amazon S3 bucket. This offloads bundle distribution from the Leader to your designated S3 bucket. Edge Nodes will pull bundles directly from S3, minimizing Leader strain.

You can configure bundling in S3 through the following ways:

- **UI**: Configure the Leader settings on the Leader Node. In the sidebar, select **Settings**. In the **Global** tab, select **System** > **Distributed Settings** > **Leader Settings**. In the **S3 Bundle Bucket URL** field, specify an S3 bucket for remote bundle storage. Format: `s3://${bucket}`.
- **YAML**: In the `instance.yml` config file, specify the S3 bucket in `master.configBundles.remoteUrl`.
- **Environment Variable**: Use the `CRIBL_DIST_LEADER_BUNDLE_URL` environment variable.

Optimized Bundle Delivery from S3

Cribl prioritizes efficient updates for Edge Nodes, regardless of their network environment. Here's how:

- **Direct S3 Downloads (Preferred)**: Cribl prioritizes downloading upgrade packages directly from Amazon S3 (S3) whenever possible, ensuring efficient updates for Edge Nodes connected to Cribl.Cloud or an on-premise Leader. The source is specified in the Leader's **Global Settings** > **Upgrades** > **Package source** setting.
- **Automatic Failover**: If the S3 download fails, a built-in failover automatically retrieves the bundle from the Leader itself (port `4200`). This redundancy guarantees updates even in challenging network environments, including private networks with restricted internet access. For these networks, Cribl prioritizes a CDN download before reverting to the Leader Node as a final resort.
- **Allowlist S3 Access (Optional)**: To further optimize downloads for Edge Nodes with internet access, consider allowlisting the regional Amazon S3 endpoint used by your bundle bucket in your firewall or proxy configuration. For example, endpoints look like `s3.eu-central-1.amazonaws.com` or `s3.us-west-2.amazonaws.com`, and dual-stack variants look like `s3.dualstack.us-west-2.amazonaws.com` or `bucket-name.s3.dualstack.us-west-2.amazonaws.com`.

S3 endpoints are regional and do not share a single wildcard hostname, so patterns such as `*.s3.amazonaws.com` are not sufficient for firewall or proxy allowlists. Refer to the AWS documentation: Using Amazon S3 dual-stack endpoints for the complete and current list of S3 regional and dual-stack endpoints.

Authenticating to S3

When storing configuration bundles in an Amazon S3 bucket, Cribl Edge needs appropriate credentials to access the bucket. There are several ways to provide these credentials, with IAM roles being the most secure and recommended approach.

IAM Roles (Recommended)

The best practice is to configure an IAM role with read access to your S3 bucket and attach this role to the EC2 instance (or other AWS service) where your Leader and Edge Nodes are running. Cribl Edge will automatically use the instance's IAM role to authenticate to S3. This method avoids the need to manage AWS credentials directly within Cribl and is the most secure option.

To authenticate via IAM roles:

1. Create an IAM role with the `AmazonS3ReadOnlyAccess` policy (or a custom policy with more restrictive permissions if needed) for the S3 bucket containing your bundles.
2. Attach this IAM role to the EC2 instance(s) running your Leader and Worker Nodes.
3. Provide the path to your S3 bucket using the CRIBL_DIST_LEADER_BUNDLE_URL environment variable or the S3 Bundle Bucket URL field in the **Distributed/ Leader** settings (For example, `s3://your-bucket-name/path/to/bundles`). `path/to/bundles` is optional.

AWS Access Keys

Alternatively, you can provide AWS access keys directly. Ensure the environment in which Cribl Edge runs in contains the following variables:

```
export AWS_ACCESS_KEY_ID="YOUR_ACCESS_KEY_ID"
export AWS_SECRET_ACCESS_KEY="YOUR_SECRET_ACCESS_KEY"
export CRIBL_DIST_LEADER_BUNDLE_URL="s3://your-bucket-name/path/to/bundles"
```

Copy

Encryption and Backward Compatibility

If your Edge Nodes are running versions older than 4.9.0 and your Leader is v.4.9.0 or newer, deployments will always use Leader-hosted bundles. This happens because of how encryption works:

- Newer Leaders encrypt bundles when uploaded to S3 to enhance security and prevent the exposure of sensitive information like secret keys.

- Older Edge Nodes check the code for errors by comparing its checksum (a unique identifier).
- Since the Edge Nodes check the encrypted code, they will always find errors because the checksum changes after encryption.

You might see an error similar to the one below in your logs: `Checksum mismatch, expected=f781b2bace2e1869e840b0ee200b57378ecae985, found=a59af697c140521ea7f42af7630ac1fce88e4ff0"**`

7.9. Leader High Availability/Failover

To handle unexpected outages in on-prem Distributed deployments, Cribl Edge supports configuring standby Leaders for failover. In this High Availability (HA) scenario, if the primary Leader goes down, one of the other Leaders, called standby Leaders, take its place. This ensures continuity of operation, including functions that require the Leader, such as Collectors and Collector-based Sources, which can ingest data without interruption.

Cribl.Cloud handles High Availability and configuring standby Leaders automatically, requiring no configuration or action on your part.



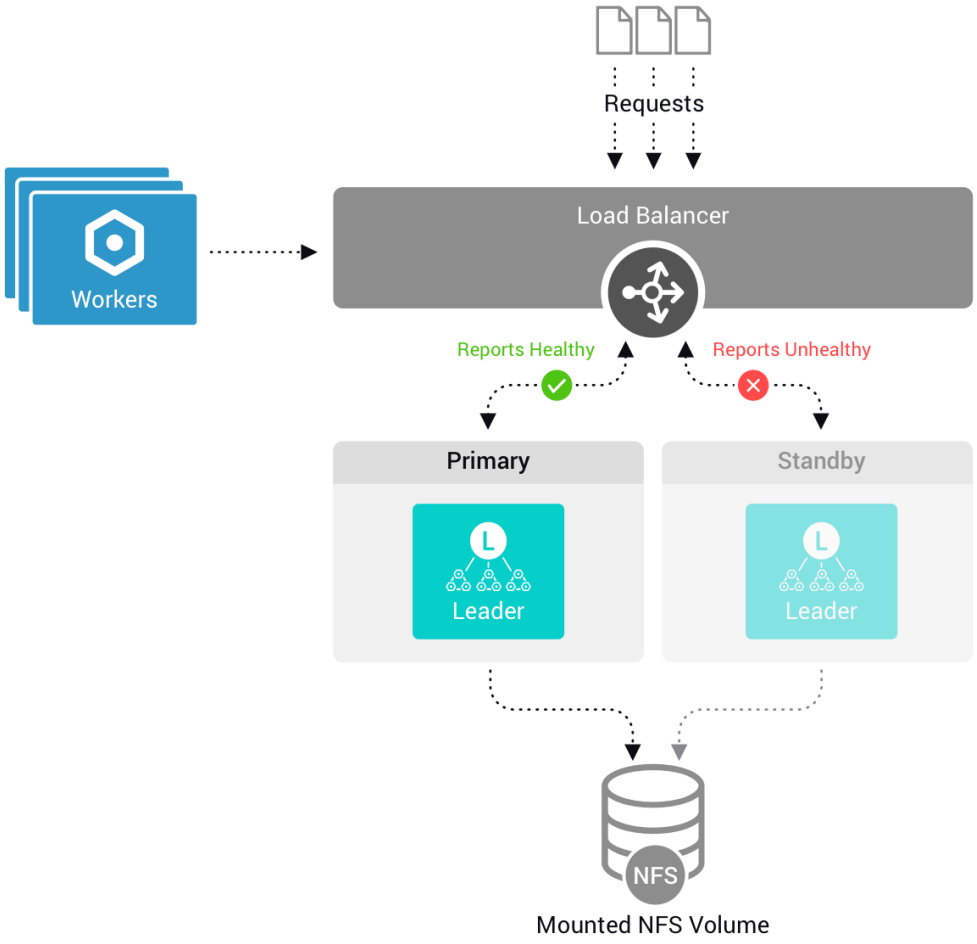
For license tiers that support configuring standby Leaders, see Cribl Pricing.

Primary and Standby Leaders

Only one Leader Node can be active at any given time, typically the primary Leader. A standby Leader (or Leaders) will become active only in the event of failover, when the primary Leader becomes unavailable.

The primary Leader stores its configurations and its git repository on the local disk. All changes to configurations and git commits are replicated to a shared failover Network File System (NFS) volume, and from it to the standby Leaders.

If the primary Leader becomes unavailable, a standby Leader takes its place. The standby then pulls the required configuration and the latest git commits from the failover volume. All Edge Nodes will connect to the new primary Leader, which retains the state and metrics of the old primary Leader



Leader High Availability/Failover Design

Leader Settings in High Availability Setups

When you first configure Leader High Availability, Cribl Edge will create a new `leader.yml` file in the local `$CRIBL_HOME/local/cribl` directory and will upload it to the failover volume. Configuration stored in the

7.9.1. High Availability Requirements

Before adding a standby Leader, ensure that you fulfill all requirements in this section.

Disk Space Requirements

Ensure that all Leaders, both the primary and the standby ones, have enough disk space available. You need enough space to hold the contents of the failover volume (except the `state` directory), all your configuration files, plus at least 10 GB in reserve for future use.

Leader Health Check Timeouts

Configure adequate timeout for Leader health checks to ensure the Leader has enough time to pull all the configuration and the git repository from the failover volume. Too low timeout could results in the Leader being terminated prematurely.

The recommended timeout scales with the number of Worker Groups and Fleets the Leader manages. The exact value depends on the size of the git repository, size of the `groups` directory, and the total number of configuration files. Approximate recommendations are:

- 2 minutes for smaller deployments
- 5 minutes for deployments with > 50 Worker Groups/Fleets
- 10 minutes for deployments with > 100 Worker Groups/Fleets

Git Configuration Requirements

If you have git timeout configured in `local/cribl.yml`, ensure that it is equal or greater than the system default of 10 minutes.

Auth Token Requirements

All Leaders must have matching auth tokens. If you configure a custom **Auth token**, make sure that all Leaders have that same token.

- In Cribl Edge, check that the values of **Settings** > **Global** > **Distributed Settings** > **Leader Settings** > **Auth token** match for each Leader.
- Or, from the filesystem, check that the values in instance.yml > `master` section > `authToken` values match for each Leader.

See How to Secure the Auth Token for the Leader Node for information on changing your auth token and allowed characters.

NFS Requirements

- On all Leader Nodes, use the latest version of the NFS client. **NFSv4 is required.**
- Ensure that the NFS volume has at least 100 GB available disk space.
- Ensure that the NFS volume's IOPS (Input/Output Operations per Second) is ≥ 200. (Lower IOPS values can cause excessive latency.)
- Ensure that ping/latency between the Leader Nodes and NFS is < 50 ms.
- Ensure your NFS system supports updating `mtime`.



You can validate the NFS latency using a tool like `ioping`. Navigate to the NFS mount, and enter the following command:

```
ioping .
```

Copy

For details on this particular option, see the ioping docs.

NFS Mount Option Requirements

The Leader Node will access large numbers of files whenever you use the UI or deploy configurations to Edge Nodes. When this happens, NFS's default behavior is to synchronize access time updates for those files, often across multiple availability zones and/or regions. To avoid the problematic latency that this can introduce, Cribl recommends that you add one of the following NFS mount options:

1. `relatime`: Update the access time only if it is more than 24 hours ago, or if the file is being created or modified. This allows you to track general file usage without introducing significant latency in Cribl Edge. To do the same for folders, add the `reldiratime` option.
2. `noatime`: Never update the access time. (Cribl Edge does not need access times to be updated to operate correctly.) This is the most performant option - but you will be unable to see which files are being accessed. To do the same for folders, add the `nodiratime` option.

Recommended NFS Configuration

If you are on AWS, we recommend the following:

- Use Amazon's Elastic File System (AWS EFS) for your NFS storage.
- Ensure that the user running Cribl Edge has read/write access to the mount point.
- Configure the EFS **Throughput mode** to `Enhanced` > `Elastic`.
- For details on NFS mount options, see Recommended NFS mount options.

For best performance, place your Leader Nodes in the same geographic region as the NFS storage. If the Leader and NFS are distant from each other, you might run into the following issues:

- Latency in UI and/or API access.
- Missing metrics between Leader restarts.
- Slower performance on data Collectors.

Load Balancer Requirements

Configure all Leaders behind a load balancer.

- Port `4200` must be exposed via a network load balancer.
- Port `9000` can be exposed via an application load balancer or network load balancer.

Active-Active (Proxy) Mode for HA Leaders

Both primary and standby Leaders use an Active-Active (Proxy) configuration to simplify load balancer (LB) routing. All Leaders can receive UI/API traffic on port `9000`, with the standby Leader automatically proxying any incoming requests to the elected primary. This architecture maintains crucial single-writer semantics, as only the primary Leader owns the state.

For proxy mode to work, the standby must be able to reach the primary over port `9000` to forward UI/API traffic. The control-plane connections on port `4200` are also proxied. Ensure firewall rules allow Leader-to-Leader communication on these paths.

Load Balancer Health Checks

Health checks over HTTP/HTTPS via the `/health` endpoint are only supported on port `9000`. Load balancers that support such health checks include:

- Amazon Web Services (AWS) Network Load Balancer (NLB). Suitable for TCP, UDP, and TLS traffic.
- AWS Application Load Balancer (ALB). Application-aware, suitable for HTTP/HTTPS traffic.
- HAProxy.
- NGINX Plus.

To ensure reliable load balancing for your Leader Nodes, configure health checks against the `/health` endpoint. For optimal performance and to avoid potential throttling, adhere to the following guidelines:

- **Polling frequency**: Set your load balancer to check the health endpoint every `60` seconds. This interval aligns with Cribl.Cloud's internal monitoring and prevents excessive API calls.
- **Avoid overly frequent checks**: Polling more often than every `60` seconds (for example, every `5` seconds) can overload the Leader's API Process.
- **Load balancer routing**: Configure your load balancer to direct traffic exclusively to Leader Nodes that return a `200` status code from the health check.

For detailed information on the `/health` endpoint query and response formats, refer to the Query the Health Endpoint documentation.

> **AWS Network Load Balancers**
>
> If you need to access the same target through a Network Load Balancer, use an IP-based target group and deactivate client IP preservation. For details, see:
>
> - Why can an instance in a target group not reach itself via NLB?
> - Why can't a target behind my Network Load Balancer connect to its own Network Load Balancer?

Source-Level Health Checks

For many HTTP-based Sources, you can enable a Source-level health check endpoint in the **Advanced Settings** tab. Load balancers can send periodic test requests to these endpoints, and a `200 OK` response indicates that the Source is healthy.

Frequent requests to Source-level health check endpoints can trigger throttling settings. In such cases, Cribl will return a `503 Service Unavailable` response, which can be misinterpreted as a service failure. A `503` response may indicate that the health checks are running too frequently instead of an actual service failure.

7.9.2. Configure Standby Leader Nodes

To prepare your deployment for High Availability, in the configuration for all Leaders, both primary and standby, you must specify the same failover volume - a shared Network File System (NFS) volume.

You can configure standby Leader Nodes in the following ways. These configuration options are similar to configuring the primary Leader Node:

- Using the UI
- Updating the YAML config file
- Using the Command Line
- Using Environment Variables

Remember, the `$CRIBL_VOLUME_DIR` environment variable overrides `$CRIBL_HOME`.

Use the UI

For all Leader Nodes, both primary and standby, configure the following:

1. In **Settings** > **Global** > **Distributed Settings** > **General Settings**, select **Mode**: `Leader`.
2. Next, on the **Leader Settings** left tab, select **Resiliency**: `Failover`. This exposes several additional fields.
3. In the **Failover volume** field, enter the NFS directory that the configurations and git commits will be replicated to. This directory must be outside of `$CRIBL_HOME`. One valid solution is to use `CRIBL_DIST_MASTER_FAILOVER_VOLUME=<shared_dir>`. See Using Environment Variables for more information.
4. Optionally, adjust the **Lease refresh period** setting, which determines how often the primary Leader tries to refresh its hold on the Lease file, and the **Missed refresh limit**, which determines how many Lease refresh periods elapse before standby Nodes attempt to promote themselves to primary.
   Increase these two values to avoid unnecessary failovers if your NFS has higher latency or is occasionally slow.
5. Select **Save** to restart.

When you save the **Resiliency**: `Failover` setting, further **Distributed Settings** changes via the UI will lock for both the primary and backup Leader. (This prevents errors in bootstrapping Workers due to incomplete token synchronization between the two Leaders.) However, you can still update each Leader's distributed settings by modifying its configuration files.

Use the YAML Config File

For all Leader Nodes, both primary and standby, configure the following in `$CRIBL_HOME/local/_system/instance.yml`, under the `distributed` section:

1. Set `resiliency` to `failover`.
2. Specify a volume for the NFS disk to automatically add to the Leader Failover cluster and trigger automated data migration.

$CRIBL_HOME/local/_system/instance.yml

```
distributed:
  mode: master
  master:
    host: <IP or 0.0.0.0>
    port: 4200
    resiliency: failover
    failover:
      volume: /path/to/nfs
```

Copy



Note that `instance.yml` configs are local, not on the shared NFS volume.

Use the Command Line

For all Leader Nodes, both primary and standby, use a command of this form:

```
./cribl mode-master -r failover -v /tmp/shared
```

For all options, see the CLI Reference.

Use Environment Variables

For all Leader Nodes, both primary and standby, configure environment variables listed in Environment Variables Reference: Adding Fallback Leaders.

Address HA Migration Timeouts

When enabling Leader High Availability (HA) on a production system with a lot of data, or when booting up new Leader Nodes, the process of moving data can sometimes take longer than expected. If the system runs out of time during this move, it might only copy part of the data, which can cause problems when the system restarts.

To mitigate the risk of incomplete data transfers, temporarily increase the system's transfer time prior to enabling HA. You can do this by adjusting a setting called `TimeoutSec` in the `cribl.service` configuration file. Locate the **Service** section and add or modify the `TimeoutSec` directive. For example, set it to `600` seconds (10 minutes):

```
[Service]
TimeoutSec=600
```

Copy

A successful data transfer will be confirmed by specific messages in the system's log, such as:

```
{"time":"2025-02-
17T22:01:46.295Z","cid":"api","channel":"ResiliencyConfigs","level":"info","message":"fa
migration is complete"}
```

A failed migration due to timeout might show logs ending with a system shutdown, like this:

```
{"time":"2025-02-
06T15:29:15.416Z","cid":"api","channel":"ShutdownMgr","level":"info","message":"Starting
shutdown ...","reason":"Got SIGTERM","timeout":0}
```



After the HA setup is complete, revert the `TimeoutSec` setting to its original value. Leaving the timeout extended indefinitely can lead to prolonged Leader downtime if a problem occurs during a future failover. Make sure to document the original setting before making changes to facilitate easy reversion.

7.9.3. Manage High Availability Deployment

Manage your High Availability deployment by monitoring Leaders, performing an upgrade, and disabling standby Leader Nodes.

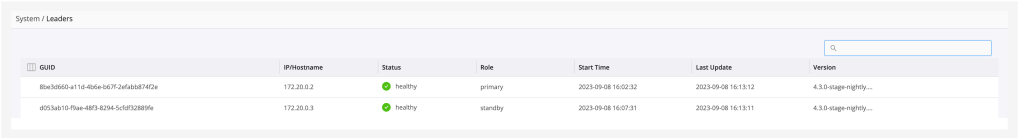
Monitor Primary and Standby Leader Nodes

To view the status of your Leader Nodes:

1. On the top bar, select **Products**, and then select **Stream**.
2. In the sidebar, select **Monitoring**, then **System**, then **Leaders**.
3. In the **Leaders** table, you can see the status of all your Leader, including the roles they play (primary or standby).



This page is only accessible when you have a standby Leader configured.



Monitoring Leader Nodes

Upgrade a High Availability Deployment



Upgrading the Leader Node through the UI is not supported for Distributed deployments with a standby Leader. Instead, use the command line (CLI) to upgrade.

Follow this upgrade order:

1. Stop the standby Leader first.
2. Upgrade the standby Leader.
3. Start the standby Leader.
4. Stop the primary Leader; the upgraded standby will take over automatically through the HA failover mechanism.
5. Upgrade the now-stopped former primary Leader for your Stream or Edge deployment.

6. Start the former primary Leader. It will rejoin and assume the standby role (since the other Leader is currently active).
7. Upgrade Edge Nodes in rolling batches; do not upgrade them to a version newer than the Leader.

Disable a Standby Leader

Cribl recommends that you maintain a standby Leader to ensure continuity in your on-prem Distributed deployment. Should you decide to disable it, contact support to assist you.

7.10. Anti-Virus Exceptions

Configure your antivirus to prevent accidental blockage of Cribl Edge.

If you are running anti-virus software on an Edge instance's host OS, here are general guidelines for minimizing accidental blockage of Edge's normal operation.

Your overall goals are to prevent the anti-virus software from locking any files while Edge needs to write to them, and from triggering any changes that Edge would detect as needing to be committed.

First, if persistent queues are enabled on any Destinations, exclude any directories that these Destinations write to. This is especially relevant if you're writing queues to any custom locations outside of `$CRIBL_HOME`.

Next, for any non-streaming Destinations that you've configured, exclude their staging paths.

Next, exclude these subdirectories of `$CRIBL_HOME`:

- `state/`
- `log/`
- `.git/` (usually only exists on Leader Nodes)
- `groups/` (on Leader Nodes)
- `local/` (on Edge or Leader)

Finally, avoid scanning any processes. Except for the queueing/staging directories already listed above, Edge runs everything in memory, so scanning process memory will slow down Edge's processing and reduce throughput.

Stop Endpoint Security from Blocking Edge on Windows

Antivirus or other endpoint security tools on Windows (such as Windows Defender) may block the `cribl` process.

To tell Windows Defender to exclude the Cribl Edge process from its security scan, you can run these commands on the appropriate Windows Edge Nodes:

```
Add-MpPreference -ExclusionProcess "C:\Program Files\Cribl\bin\nssm.exe"
Add-MpPreference -ExclusionProcess "C:\Program Files\Cribl\bin\cribl.exe"
Add-MpPreference -ExclusionPath "C:\ProgramData\Cribl"
```

Copy

7.11. Ports

Learn which ports need to be open for Cribl Edge and its integrations to function.

Cribl Edge requires certain ports to be open, and additional ports are needed if you intend to use specific integrations or options to work.

Leader

In a Distributed deployment, the following ports must be open on the Leader Node. Ensure that the Leader is reachable on those ports from **all** Edge Nodes.

Default Port	Protocol	Purpose	Direction
`9000`	HTTP/S	Cribl Edge UI.	In
`9000`	HTTP/S	Bootstrapping Fleets from Leader (on-prem).	In
`443`	HTTP/S	Bootstrapping Fleets from Leader (Cribl.Cloud).	In
`4200`	TCP	Heartbeat/Metrics/Leader requests/notifications to clients (for example: live captures, teleporting, status updates, config bundle notifications, and so on). This traffic is separate from the Notifications feature and its external Notification targets such as email, PagerDuty, Slack, webhooks, or AWS SNS.	In
`4200`	HTTP/S	Software upgrade (via path, not CDN).	In

External Notification targets: If you configure external Notification targets (for example, email, PagerDuty, webhooks, Slack, or AWS SNS), the Leader must be able to initiate outbound network connections to those services. Ensure your firewall rules allow outbound traffic from the Leader to each target endpoint over the ports and protocols required by that service (commonly HTTPS/443, or SMTP ports for email). Because these ports are target-specific, they are not listed individually here.

Outpost

The following port is used by Cribl Outpost.

Default Port	Protocol	Purpose	Direction
`9000`	TCP	Cribl Outpost UI. Login is disabled, but the Health endpoint is available.	In
`4200`	TCP	Communication between the Outpost and the Leader.	Out

Edge Nodes

The following ports are used by Edge Nodes.

Default Port	Protocol	Purpose	Direction
`9420`	TCP	Cribl Edge UI.	In
`9000`	HTTP/S	Communication with the Leader for bootstrapping (on-prem).	Out
`443`	HTTP/S	Communication with the Leader for bootstrapping (hybrid deployment), and with `https://cdn.cribl.io` to download configurations from CDN.	Out
`4200`	TCP	Heartbeat/Metrics/Leader requests/notifications to clients (for example: live captures, teleporting, status updates, config bundle notifications, and so on).	Out
`4200`	HTTP/S	Config bundle downloads from the Leader.	Out

Other Ports

Common Ports

Default Port	Protocol	Purpose	Direction
53	UDP	DNS lookups.	Out
389	TCP	LDAP Auth (non-TLS).	Out
443	HTTP/S	OIDC Auth (TLS).	Out
636	TCP	LDAP Auth (TLS).	Out

Integrations and Apps

Integrations with specific services via Sources and Destinations or apps may require opening dedicated ports on Edge Nodes.

The defaults are listed below, but when configuring each Source or Destination you can choose another port.

Default Port	Protocol	Purpose	Direction
162	UDP	SNMP Trap collection (non-TLS). The preconfigured SNMP Trap Source listens on port 9162.	In
162	UDP	SNMP Trap Destination (non-TLS).	Out
443	HTTP/S	Collection from and output to multiple HTTPS-based Sources and Destinations.	In / Out
4317	TCP	Collection from OpenTelemetry (gRPC).	In
4318	TCP	Collection from OpenTelemetry (HTTP).	In
5986	HTTP/S	Windows Event Forwarder Source.	In
8081	TCP	Kafka Schema Registry.	Out
8088	TCP	Splunk HEC input and output.	In/Out
8089	TCP	Splunk Search.	In
8125	TCP/UDP	Output to StatsD, StatsD Extended, and Graphite (non-TLS).	Out
9090	TCP	Collection/discovery from Prometheus Scraper.	Out
9092	TCP	Output to Confluent Cloud or Kafka, used when no port is provided.	Out
9093	TCP	Output to Azure Event Hubs.	Out
9200	HTTP/S	Elasticsearch API Source.	In
9514	TCP/UDP	Syslog Source.	In
9997	TCP	Splunk TCP Source.	Out
10060	TCP	TCP (Raw) data.	In
10070	TCP	TCP JSON data.	In

Default Port	Protocol	Purpose	Direction
`10080`	TCP	Collection from HTTP JSON Sources.	In
`10200`	HTTP/S	Cribl HTTP Destination.	In
`10300`	TCP	Cribl TCP Destination.	In

Cribl.Cloud

Cribl.Cloud provides a set of ports linked to Sources enabled by default for your Workspace. To view them:

1. From your Cribl.Cloud Organization's top bar, select **Products**.
2. Then from the sidebar, select **Cribl** > **Workspace**, and then **Data Sources**.

Additionally, Cribl.Cloud makes the `20000` - `20010` port range available for configuring other Sources.

Available Ports	Protocol	Purpose	Direction
`443`	TCP	Mapped to `10443` internally. Pre-configured for Amazon Data Firehose, but can be reused for other TCP-based services.	In
`20000` - `20010`	TCP	Additional Sources in Cribl.Cloud.	In

Reusing Port 443 for Other Services

External port `443` is transparently mapped by the load balancer to internal port `10443` on the Worker Group. While this port is pre-configured for Amazon Data Firehose by default, you can reuse it for other TCP-based services that require port `443` (such as DocuSign).

To use port `443` for a different service:

1. Configure a Source to listen on port `10443` internally.
2. External clients connect to your Cribl.Cloud endpoint on port `443`, which the load balancer routes to port `10443` on the Worker.

If you're already using Amazon Data Firehose on this port, you can still receive data from other services by differentiating traffic using a pre-processing Pipeline. For example, check for the presence of `__firehose*` fields or inspect `__headers` to identify the data source.

No other custom ports can be opened for Cribl-managed Worker Groups in Cribl.Cloud beyond port `443` and ports `20000–20010`.

Cribl Copilot Port

To use Cribl Copilot, your Cribl Edge deployment must be able to establish a connection to `ai.cribl.cloud`

7.12. Configure Environment Variables

Configure environment variables to control Cribl Edge behavior.

The procedure for configuring environment variables depends on the operating system the Cribl Edge Node is deployed to.

Whenever you change environment variables, you must restart Cribl Edge on the affected Nodes.



For a full list of available variables, see Environment Variables.

Configure Environment Variables on Linux

Set Variables with systemd

If Cribl Edge runs as a systemd service, add environment variables to the systemd override file (for example, `/etc/systemd/system/cribl-edge.service.d/override.conf`) so they persist across restarts and upgrades.

1. Open the override file for editing:

```
sudo systemctl edit cribl-edge
```

Copy

2. Add the variables under the `[Service]` section:

override.conf

```
[Service]
Environment=CRIBL_NOEXEC=1
Environment=HTTP_PROXY=http://proxy.example.com:3128
```

Copy

3. Reload the systemd daemon and restart Cribl Edge:

```
sudo systemctl daemon-reload
sudo systemctl restart cribl-edge
```

Copy

For details on persisting overrides, see Persisting Overrides.

Set Variables Without systemd

If you are not using systemd, add `export` lines to a shell profile file (such as `/etc/environment` or the Cribl user's `~/.bashrc`) so they load on login:

```
export CRIBL_NOEXEC=1
export HTTP_PROXY=http://proxy.example.com:3128
```

Copy

Configure Environment Variables on Windows

Set Variables During Installation

You can configure environment variables while installing Cribl Edge on Windows using either the interactive wizard or a parameter for the MSI installer.

- **Wizard**: In the setup wizard, in the **Set Environment Variables** modal, enter one variable per line. See Use the Wizard for more details.
- **MSI installer**: When installing via MSI, use the `SETENVVARS` parameter with pipe-delimited key-value pairs. The entire string must be quoted:

```
msiexec /qn /i cribl-<version>-<build>.msi ^
  SETENVVARS="HTTP_PROXY=http://203.0.113.1:8080|HTTPS_PROXY=http://203.0.113.1:
  MODE=mode-managed-edge ^
  HOSTNAME=<yourhostname> ^
  PORT=4200 ^
  AUTH=myAuthToken ^
  FLEET=myfleet
```

Copy

See Use the MSI Installer for the full syntax.

Set Variables After Installation

Environment variables specified during installation are stored in the Windows registry under `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\cribl` under the `Environment` key.

To add or modify variables after installation, edit this registry key directly. See About Environment Variables for details.

Restart the Cribl Edge service for the changes to take effect.

Configure Environment Variables on macOS

Add Variables to the LaunchDaemon Plist

To define environment variables to a macOS deployment of Cribl Edge, you need to add them to the `io.cribl.plist` file.

1. Open the Cribl plist file, for example: `sudo vi /Library/LaunchDaemons/io.cribl.plist`.
2. Locate the `EnvironmentVariables` key and add your variable under it. For example, to define the `NODE_EXTRA_CA_CERTS` variable, add:

```
<key>EnvironmentVariables</key>
<dict>
   <key>NODE_EXTRA_CA_CERTS</key>
   <string>path/to/cert/ca.pem</string>
   <!-- other keys -->
</dict>
```

Copy

3. Reload the plist file by running:

```
sudo launchctl unload /Library/LaunchDaemons/io.cribl.plist
sudo launchctl load /Library/LaunchDaemons/io.cribl.plist
```

Copy

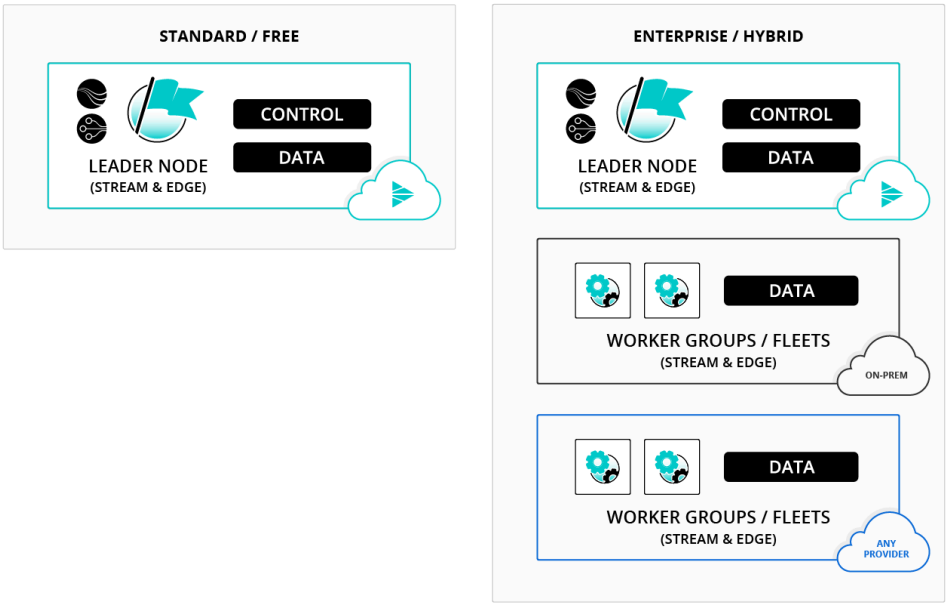
The environment variable added this way is retained when upgrading the Edge Node in a Fleet. However, if you upgrade via the bootstrap script, the plist file is replaced. In that case, re-add the environment variable.

7.13. About Cribl.Cloud

In a Cribl.Cloud deployment, Cribl assumes responsibility for hosting and managing your Cribl Edge infrastructure.

Cribl.Cloud comes in both free and paid versions. A free Cribl.Cloud account allows up to 1 TB/day of data throughput (data ingress + egress), but does not have the option to have multiple Fleets. By upgrading to a Cribl.Cloud Enterprise plan, you can implement a hybrid deployment of any complexity.

In hybrid deployments, the Leader resides in Cribl.Cloud, and manages Stream Workers and Edge Nodes regardless of their location. Your data processing configuration can consist of any combination of Workers and Edge Nodes: Cribl-managed, in private cloud instances that you manage, or in your data centers.



Standard/free versus Enterprise/hybrid deployment

For a comparison of features an capabilities of Cribl.Cloud vs an on-prem deployment, see Cribl.Cloud vs. Self-Hosted.

For an overview of additional features available on Enterprise plans, see Pricing.

If you're new to Cribl Edge, please see Cribl Edge Basic Concepts and the Getting Started Guide for help onboarding. Cribl.Cloud always runs in Distributed mode - for details about this, see Simplified Distributed Architecture.

Cloud Pricing

7.13.1. Register Cribl.Cloud Organization

To get started with Cribl.Cloud, sign up on the Cribl.Cloud portal, to create your Cribl.Cloud Organization.

Your Organization will have a dedicated portal with its own network and access boundaries, isolating your Cribl resources from other users. For more information, see Cribl.Cloud Data Isolation.

By default, your instance of Cribl.Cloud will be on the **free** Cribl.Cloud plan, which includes certain throughput and administration limits. When you need more capacity or features, you can upgrade to a paid or Enterprise plan. To upgrade, select **Go Enterprise** in your Cribl.Cloud portal.



You can find the Cribl.Cloud Suite on the AWS Marketplace. When you're ready to upgrade, you can use your Enterprise Discount Program (EDP) credits to run Cribl products, billed directly through your AWS account. There is no need for a separate procurement process.

Note that each user can register only one active Organization on the free tier. For details, see notes on Member Permissions.

To sign up for a free Cribl.Cloud instance and register your Organization:

1. Go to https://cribl.cloud/signup/ and create an account using your work email address.
2. Follow the instructions to set up your Organization, select its region for your Cribl.Cloud deployment, and to provide some information about your goals in using Cribl.

When you complete the registration steps, you will see your new Cribl.Cloud deployment home page. The home page suggests possible actions you can take based on the answers you provided during registration.

From there, consider completing the Getting Started Guide tutorial for a walk-through of the main features and common use cases for Cribl Edge.

Cribl.Cloud Organization Regions

The Organization region defines where the Leader Node resides.

Once you select a region when registering your Organization, you cannot change it. You can, however, create Cribl-managed Stream Worker Groups in different AWS and Azure regions.

In Cribl Search, the Organization region is the physical location where searches are executed. In Cribl Lake, the region defines the location where data is stored.

Troubleshooting Resources

If you get stuck during registration, Cribl University has a troubleshooting guide on How to Log Into Cribl.Cloud walks you through the registration and login workflow. To follow the direct course link, first log into your Cribl University account. Once logged in, check out other useful short courses:

- Troubleshooting Criblets
- Advanced Troubleshooting

7.13.2. Manage Cribl.Cloud Organization

The Cribl.Cloud portal is the place where you get an overview of and manage your organization, configure network settings, and control user access and billing information.

Switch Organizations

When you own or are a Member of multiple Cribl.Cloud Organizations, you can switch between the Organizations:

1. In the top bar, hover over the name of your Organization.
2. In the drop-down, select the Organization you want to switch to.

Alternatively, in the drop-down you can select **Manage Organizations**. You will move to an Organization landing page, displayed to an Organization's Owners and Admins, where you can view details of all your Organizations, switch to any of them, add a new one or delete one, if you have proper permissions.

Configure Organization Details

To customize your Organization information, in the sidebar, select **Organization**.

You can modify the following information:

Field name	Description
Organization Name	A "friendly" name for your Organization. Upon signing in, Members will see this alias above the Organization ID on the Select Organization page.
Description	Add further details about your Organization.
Opt in to beta features	If displayed, this check box enables access to new options that Cribl has not yet made generally available. As with all beta features, expect some instability in exchange for advancing to the cutting edge of your Cribl.Cloud.

Select **Save** to immediately apply your changes.

You can check additional information about your Organization, including the AWS Cloud region in which it's located, the software version, and license, by selecting **Organization Details** at the bottom of the sidebar.

Find Your Cribl.Cloud Organization Region

To locate your Cribl.Cloud region, click **Organization Details** in the sidebar of your Cribl.Cloud portal page (for Admins and Owners) or on any Cribl product home page (any user). It's helpful to know the Organization region when there is a release or maintenance upgrade, because Cribl.Cloud Organizations can use specific upgrade windows based on the Leader region. The Organization region might not match your physical location.

Ingress Ports

Your Cribl.Cloud Organization's preconfigured and user-configurable ports for inbound data are set at the Workspace level. For details, see Cribl.Cloud Ports.

Get Support

In the top bar, select the **Support** icon to access a set of links with additional resources about Cribl Edge, including:

- Sandboxes (free, interactive tutorials on fully hosted integrations)
- Cribl University
- Concept/demo videos

Refer to Cribl Status for information about Cribl.Cloud availability, including current and historical uptime data, and incidents, split by component and AWS Region.

You can also subscribe there to incident notifications via email, text, Slack, or RSS.

Manage Personal Settings

Use the **Account** menu in the top right of the Cribl.Cloud portal to configure your personal settings.

Select **My profile** to update your personal information. Here, you can also reset the configured biometric authentication. Do this, if you want to re-configure the authentication, or if you don't want to be prompted to log in through fingerprint or face recognition.

7.13.3. Cribl.Cloud Enterprise

With a Cribl.Cloud Enterprise plan, you have these options and workflow flexibility:

- Configuring and managing multiple Worker Groups and Fleets.
- Notifications within Cribl apps, to PagerDuty, and/or to other services via webhook.
- Fine-grained, role-based control of Member Permissions on your Cribl.Cloud Organization and on individual product resources.
- Single sign-on/SSO authentication from external identity providers.
- The hybrid deployment option, described just below.

With an Enterprise Plan, the Leader resides in Cribl.Cloud, and controls a flexible mix of Cribl-managed and/or customer-managed Fleets. Cribl manages the Leader's high availability on your behalf.

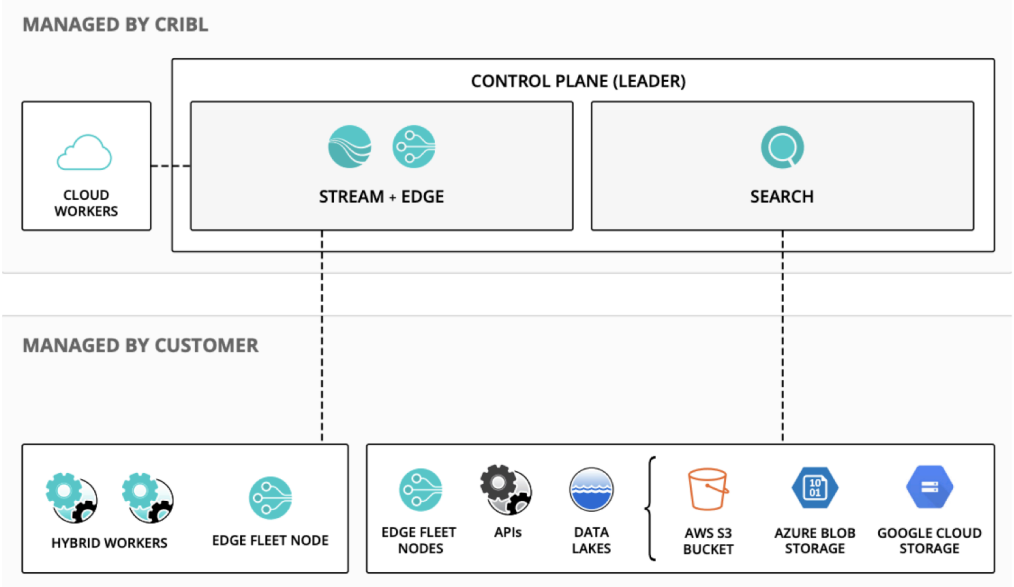


For other Enterprise features, see our Pricing page.

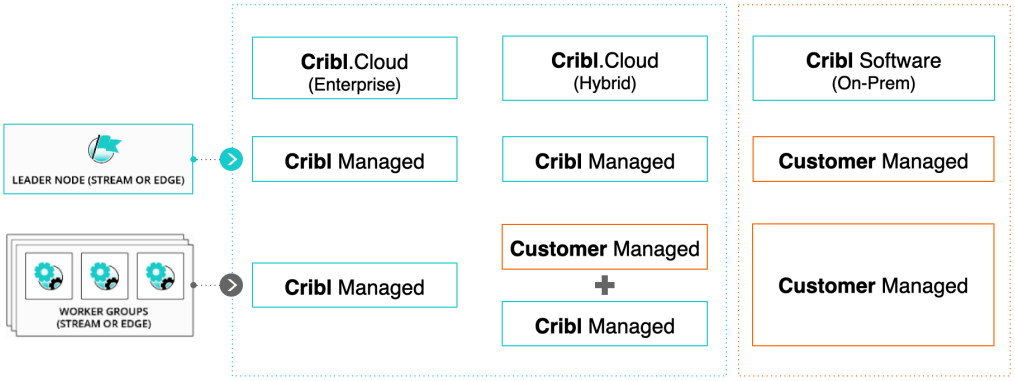
Hybrid Deployment

The diagrams below show the comparative flexibility of a hybrid Cribl.Cloud deployment. The Leader (control plane) resides in Cribl.Cloud, while the Workers that process the data can be in any combination of the following environments:

- In Cribl.Cloud, managed by Cribl.
- In public or private cloud instances that you manage.
- On-prem in your data centers.



Enterprise hybrid deployment, with control plane and Cribl-managed Workers in Cribl.Cloud



Enterprise hybrid deployment, with only control plane in Cribl.Cloud

As the footprint of your operations grows or changes, this flexibility makes it easy to reconfigure Cribl Edge in tandem. You can rapidly expand Cribl Edge observability into new cloud regions - and replace monitored hardware data centers with cloud instances - all while maintaining one centralized point of control.

You can also add Workers or Edge Nodes, and reassign them to different Fleets, by easily auto-generating Stream or Edge command-line scripts within Cribl Edge's UI.

Hybrid Requirements

A hybrid deployment imposes these configuration requirements:

- Hybrid Workers (meaning, Workers that you deploy on-prem, or in cloud instances that you yourself manage) must be assigned to a different Fleet than the Cribl-managed `default` Group - which can contain its own Edge Nodes.

- All Edge Nodes' hosts must allow outbound communication to the Cribl.Cloud Leader's port 4200 at `https://main-<Organization-name>.cribl.cloud:4200`, to enable configuration and workload management by the Leader.
- On all Edge Nodes' hosts, firewalls must allow outbound communication on port 443 to the Leader and to `https://cdn.cribl.io`. This port is also used to bootstrap hybrid Workers from the Leader.
- All Edge Nodes require connectivity to `https://cdn.cribl.io/telemetry/`. For details on testing this connectivity, on the metadata transmitted to Cribl, and on how we use that data, see Telemetry Data.
- If this traffic must go through a proxy, see System Proxy Configuration for configuration details.
- To verify your Leader's region and public URL, open the Access Details modal.

Note that you are responsible for data encryption and other security measures on Edge Node instances that you manage.

Adding (Bootstrapping) Workers

To add Workers to your hybrid Cribl.Cloud deployment, Cribl recommends that you use the script outlined in Bootstrap Workers from Leader. Hosts for the new Workers must open the same ports (4200 and 443) listed in Hybrid Requirements.

You have three options for generating the script, outlined in these subsections of the Bootstrap topic linked above:

- Auto-generate it from the Leader's UI.
- Make a `GET` API request to the Leader.
- curl the same API request.



In Cribl Edge, you access all these bootstrap options via the Manage Edge Nodes page's **Add/Update Edge Node** control.

Hybrid Cribl HTTP/TCP Configuration

If you use the Cribl HTTP Destination and Source pair, or the Cribl TCP Destination and Source pair, to relay data between Edge Nodes connected to the same Leader, configuring hybrid Workers demands particular care.

The Edge Nodes that host each pair's Destination and Source must specify exactly the same Leader Address. Otherwise, token verification will fail - breaking the connection, and preventing data flow. In hybrid Cribl.Cloud

8. Manage Fleets Of Edge Nodes

Get started with managing Fleets of Edge Nodes in a Distributed deployment.

In a distributed environment, a single Leader Node can centrally manage Edge Nodes. The Leader is responsible for keeping configurations in sync, tracking Edge Node status, and monitoring metrics.

Concepts

Single Edge Node - a single Cribl Edge instance, running as a standalone (not distributed) installation on one server/endpoint.

Leader Node - an instance running in **Leader** mode, used to centrally author configurations and monitor Edge Nodes in a distributed deployment.

Managed Edge Node - an instance running as a **Managed Edge**, whose configuration is fully managed by a Leader Node. (By default, will poll the Leader for configuration changes every 10 seconds.)

Fleet - a collection of Edge Nodes that share the same configuration. You map Nodes to a Fleet using a Mapping Ruleset.

Subfleet - a Subfleet groups and manages Edge Nodes that share the same configuration. Each Subfleet inherits configurations from its parent Fleet. Updating and deploying parent-level configurations applies the changes to the Subfleets. The Subfleets then deploy the configuration changes to their Edge Nodes.

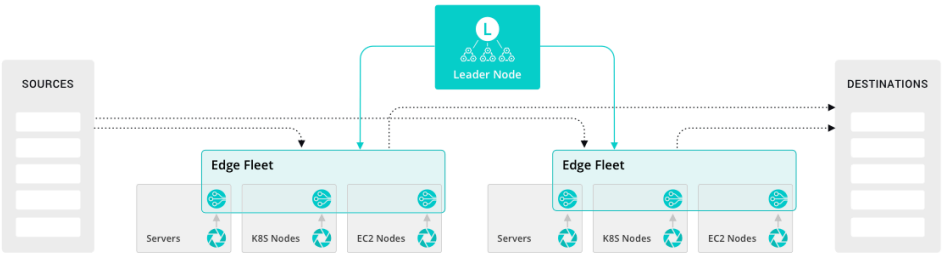
Worker Process - a Linux process within a Single Instance, or within Edge Nodes, that handles data inputs, processing, and output. The process count for Edge Nodes is constrained to 1.

Mapping Ruleset - an ordered list of filters, used to map Nodes into Fleets.

Multiple Fleets are very useful in making your configuration reflect organizational or geographic constraints. E.g., you might have a U.S. Fleet with certain TLS certificates and output settings, versus an APAC Fleet and an EMEA Fleet, each with their own distinct certs and settings.

Architecture

This is an overview of a distributed Edge deployment's basic components. (The type/s of Nodes within each Fleet will vary depending on your overall architecture:)



Distributed deployment architecture

Leader Node

- API Process - Handles all the API interactions.
- *N* Config Helpers - One process per Fleet. Helps with maintaining configs, previews, etc.

Edge Node

- Single Process - Handles everything; collection, processing and communication with the Leader Node.

Leader Node Requirements

- **OS**: Linux: RedHat, CentOS, Ubuntu, AWS Linux (64bit)
- **System**: +4 physical cores, +8GB RAM, 5GB free disk space
- **Network**:
  - Heartbeat Port: `4200`: managed Edge nodes communicate with the Leader Node on port `4200` by default.
  - UI/API Port: `9000`: users acccess the Leader on port `9000` by default.
- **Git**: `git` must be available on the Leader Node. See details below.
- **Browser Support**: The five most-recent versions of Chrome, Firefox, Safari, and Microsoft Edge.



We assume that 1 physical core is equivalent to 2 virtual/hyperthreaded CPUs (vCPUs). All quantities listed above are minimum requirements. We recommend deploying the Leader on stable, highly available infrastructure, because of its role in coordinating all Edge instances.

Edge Node Requirements

See Deployment Planning for requirements and other details.

Version Control with `git`

Leader Node requires `git` (version 1.8.3.1 or higher) to be available locally on the host.

Configuration changes must be committed to git before they're deployed.

If you don't have `git` installed, check here for details on how to get started.

The Leader node uses `git` to:

- Manage configuration versions across Fleets.
- Provide users with an audit trail of all configuration changes.
- Allow users to display diffs between current and previous config versions.

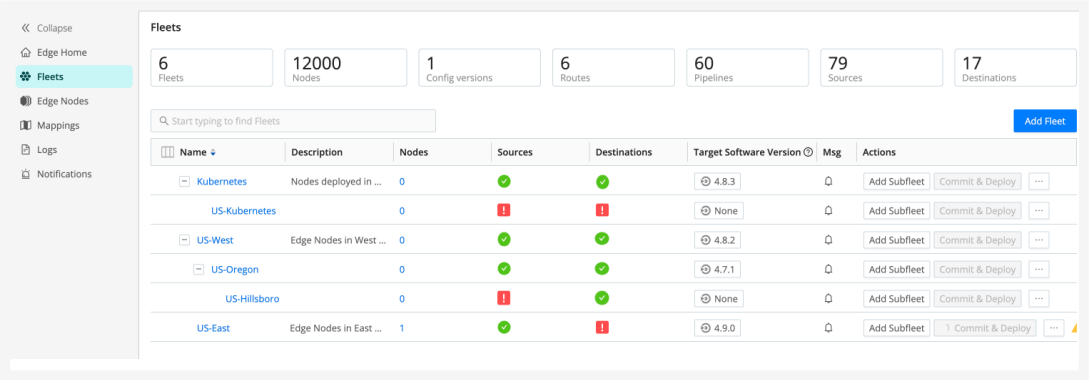
8.1. Fleets

Learn how to access your Fleets in Cribl Edge.

The **Fleets** page is available on certain plan/license tiers. For details, see Pricing.

Access the Fleets Page

In Cribl Edge, select **Fleets** from the sidebar. This view shows all of the Fleets and Subfleets you have access to in the instance or Organization.

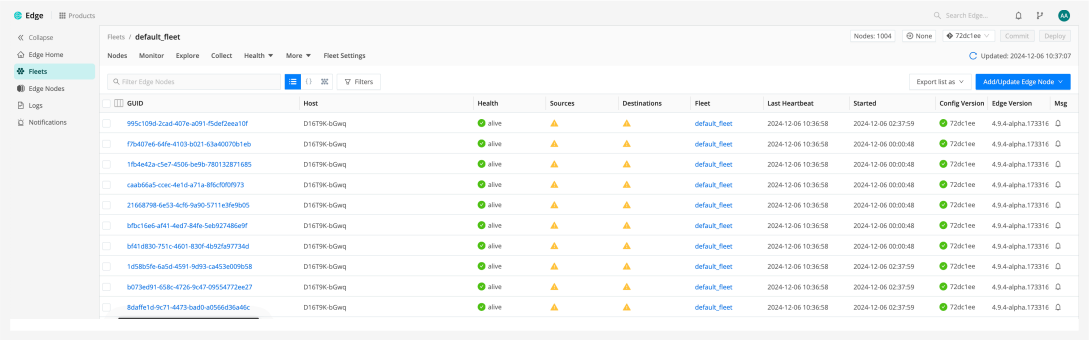


Manage Fleets

To explore more information for each of your configured Fleets, select a Fleet **Name** to open it.

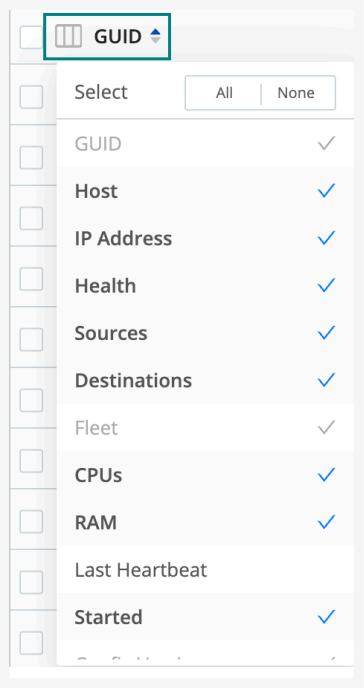
View Edge Nodes in a Fleet

When you open a Fleet, you'll land on the **Nodes** page, in **List View**.



Fleet Landing page

You can use the column selector to show and hide columns in this view. Select the column icon next to **GUID**.



Edge Nodes column selector

A few highlights on some of the options:

Field	Description
Health column	This column displays an icon indicating the overall health of the Edge Node.
Sources	Shows the individual health statuses of the Edge Node's Sources. You can sort and filter these Sources by their health status, which helps you quickly identify Nodes that are experiencing issues with a Source.
Destinations	Shows the individual health statuses of the Edge Node's Destinations. You can sort and filter these Destinations by their health status, which helps you quickly identify Nodes that are experiencing issues with a Destination.
Edge Version	Displays a status icon next to the version number. This icon can provide information about the upgrade status for a Node. For more information on upgrade status, read View the Status of Fleet Upgrades.
CPU	Shows the CPU utilization for the Edge Node. Use the column selector icon to reveal this option.
RAM	Displays the memory usage of the Edge Node. Use the column selector icon to reveal this option.

The Edge Nodes list also serves as the "transporter room," allowing you to teleport into each of the Node's interfaces.

Understand the Health Status Icons

The Source and Destination health icons represent the following statuses:

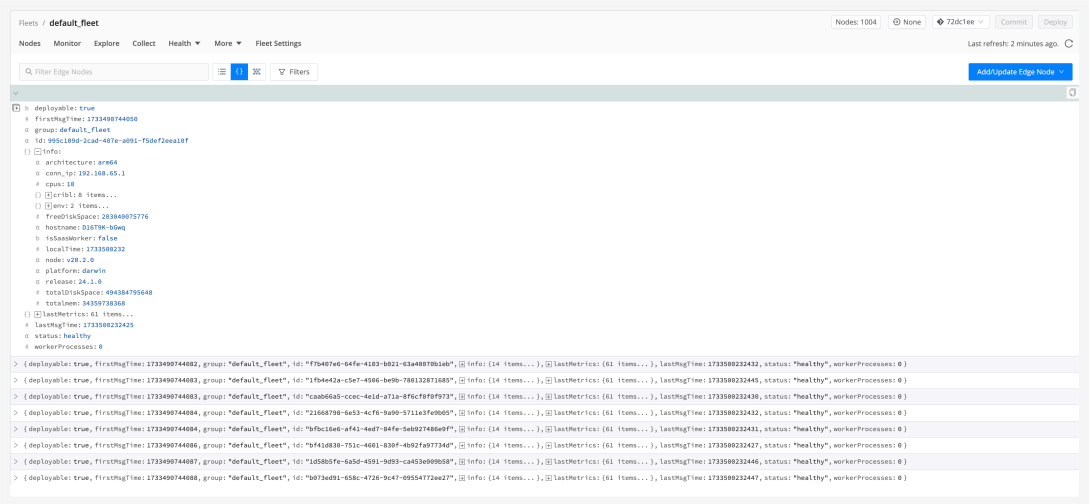
- Green checkmark: Everything is good! All Sources and Destinations are healthy.
- Yellow warning icon: Attention needed. One or more Sources or Destinations are in a warning state.
- Red exclamation point: Critical issue! One or more Sources or Destinations have encountered an error.
- Indeterminate: No data available yet for Sources or Destinations.

Ways to View Nodes

To switch between the three Edge Node views, select the List View icon, JSON View { } icon, or Map View icon.

List View: A list of all the Nodes and their statuses in the Fleet.

JSON View: Displays each Node and its status in the Fleet as an expandable JSON object. The JSON view provides inputs to create custom Node filters in **Advanced** mode. For details, see Filter Modes.

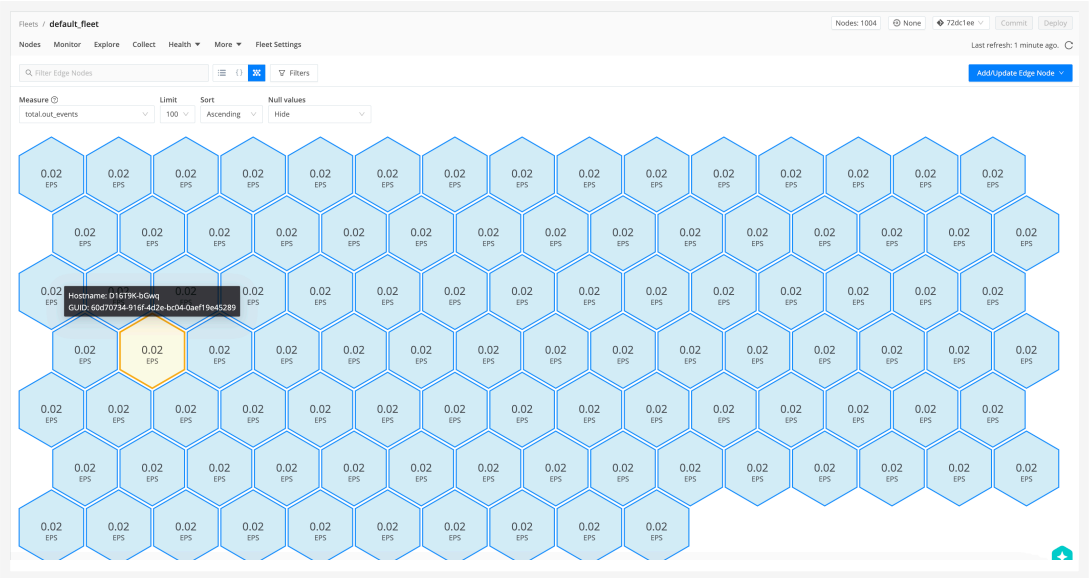


JSON View

Map View: Here, a query builder allows you to display metrics from the Edge Nodes in the Fleet.

The metrics that appear in the **Measure** list depend on the option selected in **Fleet Settings** > **Limits** > **Metrics** under **Metrics to send from Edge Nodes**. The metrics `total.in_bytes`, `total.in_events`, `total.out_bytes`, and `total.out_events` always appear in the list; the display of any other metrics depends on what the Edge Nodes are sending to the Leader. To learn more about these metrics options, see Control Metrics.

A hexagon-based map view displays the resulting metric combination for each of your Edge Nodes. Hovering over on one of the hexagons displays the **GUID** and **Hostname** of the Edge Node.



Map View

Select any of the hexagons to view the Node Info modal containing details on the Edge Node. For details on this panel, see our Tracking Data Activity for an Edge Node topic.

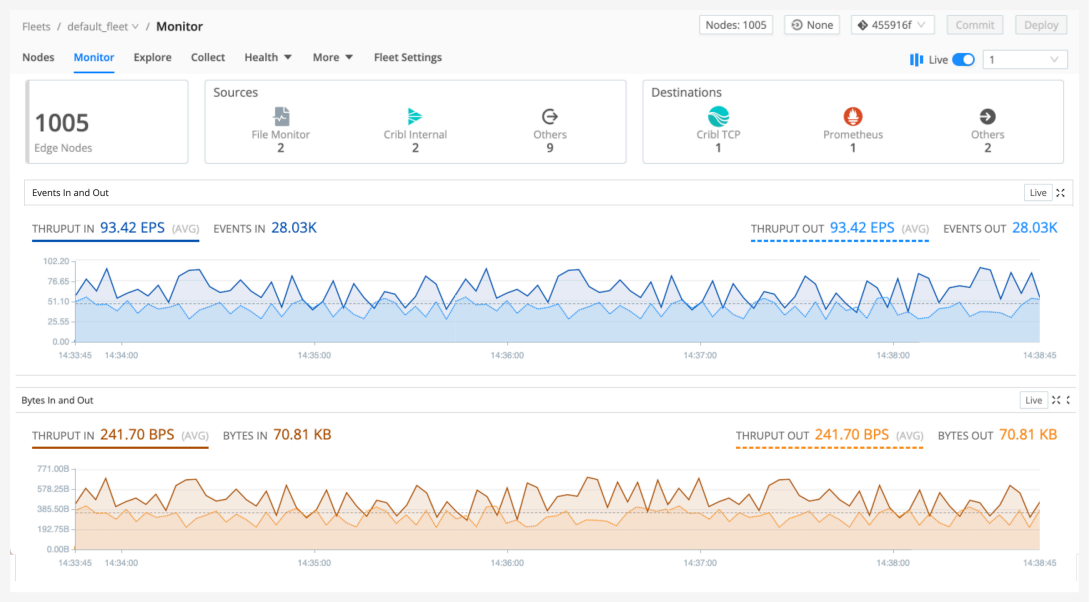
Teleport into an Edge Node

To teleport from the Leader into the Edge Node, you must turn on **Enable teleporting to Nodes** in Fleet Settings. Then, select the Edge Node link in the **GUID** column.

For more information on what you can do while teleported into an Edge Node, see Manage Edge Nodes.

Monitor Fleets

The **Monitor** tab displays information about your configured Edge Nodes. For more information about monitoring, see our topic on Monitoring Health and Metrics.



Fleets Monitor View

Here, view the following information:

Name	Description
Number of Edge Nodes	The number of Edge Nodes configured in this Fleet.
Target Version	If you have a target version selected for the Fleet, it will appear on this button. If not, the button will show **None** (the default). To learn more about this feature, see Upgrade Edge Nodes in a Fleet.
Events In	Total number of Events in the last 5 minutes of data collected. You can change the display's granularity from the default `last 5 min`, selecting a variety of time ranges from 1 min up to 1 day. (The latter covers the preceding 24 hours, and this maximum window is not configurable.)

8.1.1. Design Fleet Hierarchy

Design your Fleets into a practical hierarchy.

Cribl Edge enables you to group Edge Nodes into logical Fleets and Subfleets. A well-designed Fleet hierarchy is extremely important for the health of the Leader Node and lets you take advantage of configuration inheritance.

Fleet Design Better Practices

Always design your Fleet hierarchy before deploying your Edge Nodes. For details, see Designing Your Fleet Hierarchy.

Consider Leader Health

To ensure the health of your Leader Node, follow these guidelines:

- Consider the number of Edge Nodes restarting at once, since they will all be pulling their configurations from the Leader at the same time.
- For deployments with over 100,000 Edge Nodes, change the default metrics controls to `minimal`.

Deploy Nodes to Specific Fleets

Use the bootstrap script in the Add/Update Edge Node modal to deploy Edge Nodes to a specific Fleet. Depending on your Fleet design, Cribl recommends adding tags to the bootstrap script to uniquely identify the Fleet. You can use these tags for automatic mappings.

Use `default_fleet` as Fallback for Stray Nodes

Do not map Edge Nodes to `default_fleet`. It should serve as fallback for mapping errors.

Create mappings for all the Fleets so that Edge Nodes are automatically placed in the Fleet when they bootstrap and connect to the Leader. This way, if any rogue Edge Node gets deployed, it will automatically show up in `default_fleet`. You can then monitor `default_fleet` to catch those Nodes and reassign them.

If you disable all the Sources in the `default_fleet`, there will be no impact on data consumption.



This guidance doesn't apply to the free license, where the only option is `default_fleet`.

Design Your Fleet Hierarchy

If your organization is deploying a significant number of Edge Nodes, it's essential to thoroughly plan and design your Fleet architecture. If you are moving from other agent-based systems, avoid exactly replicating the previous structure, as Cribl Edge operates differently.

The first question you should ask is, do you need Subfleets? In some cases, a single-level, flat design is sufficient.

In more complex environments, layering the configurations can help you to manage a large number of Edge Nodes that share common yet differentiated configurations. For example, as an administrator, you might need multiple layers of configurations to manage the following: `Org` > `Department` > `OS`. In this case, the Edge Nodes at the `OS` level can inherit and share configurations from the `Department` level. You can update a parent-level configuration (Fleet) and have it applied to all Subfleets and their respective Edge Nodes, reducing the time and effort needed to manage them.

Maintaining a large number of Fleets can introduce operational complexities and increase the overhead of managing your Cribl Edge deployment. Before creating a large number of Fleets, explore whether your requirements can be effectively addressed by leveraging Cribl Stream as a central Destination and using Pipelines within a more streamlined Fleet structure to manage routing and processing logic.

Fleet Hierarchy Options

There are several ways you can build the hierarchy:

- Geographical: If your organization collects different data categories in different regions, consider this option.
- Data center: This is a good option if you maintain data centers in different regions.
- Server function (web server, database, and so on): Another good way to organize your Fleets can be based on server functions.
- OS-based: If you are running an environment with machines on different platforms, consider separating them into different Fleets per OS.

Fleet Limitations and Recommendations

When designing your Fleet hierarchy, consider the following recommendations:

- **Recommended Depth:** To promote manageability and performance, limit your Fleet hierarchy to no more than **two levels** of Subfleets below the top-level Fleet.
- **Cloud Environment Restriction:** Cribl.Cloud environments have a current limitation of **250 Fleets** per deployment. Carefully plan and optimize your Fleet design to stay within the limit.

If you are working in a large organization and find these limitations insufficient, **engage with your Cribl account team and Cribl design partners** to develop an optimized Fleet design that meets your specific needs.

Fleet Naming

Consider naming your Fleets and Subfleets in a way that helps you recognize the configurations at a glance.

For example, you can use the top-level Fleet name as part of your lower-level Fleets. Naming your Fleet with a logical `<parent-fleet>-<subfleet1>-<subfleet2>` will easily provide you with context whether you are viewing at the Fleet, Subfleet, or Edge Node level. It can also help identify Edge Nodes that are out of place or mapped incorrectly.

Here's an example of naming a Fleet with two levels of Subfleets:

- BZT
    - BZT-NewYork
        - BZT-NY-Linux
        - BZT-NY-Windows
    - BZT-LA
        - BZT-LA-K8s
        - and so on

Other guidelines to keep in mind when naming your Fleets:

- Fleet names can't have spaces in them; use a dash or underscore instead.
- You can't reuse a Fleet name.
- Once you create a Fleet, you can't edit the name.
- You can't delete a Fleet if there are Subfleets under it.

Plan Fleet Inheritance

An efficient Fleet hierarchy design lets you take full advantage of Fleet inheritance. Each Subfleet will inherit the configurations of every parent Subfleet before it. To learn more about how inheritance works, see Fleet Inheritance.

Here are some guidelines to keep in mind, as you think about layering your configurations.

Top-Level Fleets (Grandparents)

Assign the top-level Fleets for company-wide configurations. Use this level to:

- Apply certificates and other company-wide settings.
- Avoid OS-specific or site-specific configurations.

Level 1 Subfleets (Parents)

Per-site configurations. Use this level to:

- Configure site-specific Sources/Destinations.
- Specify any site-specific configurations, like common Destinations.

Level 2 Subfleets (Children)

Configurations used by groups of similar systems. Use this level to:

- Apply OS-specific configurations at this level. For example, an “All Linux” and “All Windows” for a given site.
- Group common OS-specific metrics and common log files into OS-specific Fleets.

Level 3 Subfleets (Grandchildren)

While Cribl typically doesn’t recommend creating more than two levels of Subfleets, if your specific complex organization requires them, use the third level to apply function-specific Subfleets to add specific metrics, logs, and/or Windows Event sources. For example, you can group by:

- Apache HTTPD
- Database
- SMTP servers
- Active Directory
- IIS
- Exchange

8.1.2. Fleet Inheritance

Manage Fleet configuration with inheritance.

Inheritance simplifies and speeds up Fleet setup by allowing configurations to automatically propagate to Subfleets.

When creating a new Fleet, you can indicate a parent that the new Fleet will inherit configurations from. Your new Fleet will become a Subfleet of the parent. Whenever you modify configurations in the parent, its Subfleets will receive the same changes.

To take advantage of inheritance, you can organize your Fleets and Subfleets into a hierarchy of configuration layers. Layering the configurations simplifies managing a large number of Edge Nodes that share common yet differentiated configurations. To learn more about planning Fleets and Subfleets, see Design Fleet Hierarchy.

How Inheritance Works

When you save and commit a configuration change in a Fleet, its Subfleets (and their Subfleets, that is, all its descendants) will receive the same configuration change.

You can view the configuration changes that can be inherited in the **Git Changes** modal before committing them. This view represents the files that contain configuration changes. See Configuration files for more details about those files.



When you make changes to a parent Fleet configuration, use the **Commit and Deploy with Subfleets** option. This way, you are always keeping the inherited configurations in sync.

Overriding Inheritance

If you change a configuration setting in a Subfleet, the inheritance link with the parent Fleet breaks. Any further changes to the modified setting in the parent will no longer propagate to the Subfleet and its further descendants.

To revert broken inheritance (that means, to allow a Subfleet to inherit again from its parent after inheritance has been broken), manually reset the values for a given configuration key to the same state as the parent.



Consult the **Git Changes** modal after resetting to make sure no unintended Subfleet-level configuration is left.

Sources, Destinations, Packs, Knowledge Objects, and other objects, are inherited as a whole. If you modify the configuration for an object in a Subfleet, this object will no longer inherit further changes, even in other settings.

To revert a configuration change to an object, you can delete this object in the Subfleet. The Subfleet will then inherit the configuration from its parent again.

Removing Inherited Sources or Destinations

If you delete an inherited Source or Destination from the Subfleet, the Subfleet will inherit it again from its parent Fleet. If you want to ensure a Source or Destination does not propagate again to the Subfleet, **disable** it instead of deleting.

Inherited Configurations

Subfleets inherit the following Fleet-level configurations:

- Fleet Settings
- Sources and Destinations
- Routes and Pipelines
- Packs (including local modifications to Packs installed from an external source)
- Knowledge Objects
- Data samples
- Certificates and shared Secrets



Settings configured on the **Overview** tab of **Fleet Settings**, such as **Description**, **Tags**, and **Time to keep disconnected Nodes**, are not inherited by Subfleets.

Certificate Inheritance

Certificates, configured in **Fleet Settings** > **Security** > **Certificates**, are inherited by Subfleets alongside other Fleet settings. They are stored locally on each Edge Node in the `local/edge/auth/certs/` directory.

For more information about certificates, see Secure Sources and Destinations with Certificates.

8.1.3. Configure Fleet Settings

Configure teleporting, throughput throttling, logging, and security at the Edge Fleet level.



Required Permission: **Admin**

In Cribl Edge, you can use **Fleet Settings** to configure Node teleporting, throughput throttling, logging, upgrading, and security at the Fleet level.

To edit Fleet Settings:

1. Select the Fleet, then select **Fleet Settings**.
2. Configure the following settings per Fleet.

General Settings

Fleet Configuration

You can configure the following general Fleet options on the **Overview** tab:

Fleet Configuration Setting	Description
Enable teleporting to Nodes	Toggle on to enable authenticated access to the Edge Node UI from the Leader. Toggle off to disable. For more information about teleporting, see Teleport into an Edge Node.
Description	Optionally, add or edit the Fleet description.
Tags	Use tags to organize Fleets into logical categories. Then, you can search by tags on the **View all Fleets** interface. This search filters the list of Fleets, showing only those with the tag you entered.
Time to keep disconnected Nodes	Configure how long the Fleet should retain information about a Node that disconnects from the Leader. Format examples: 8h, 5d, 1w (format defaults to days if you only specify a number). Default: 1 day.
Inherited configuration	Optionally, select an existing Fleet that this Fleet will inherit configurations from.

Settings configured on the **Overview** tab are not inherited by Subfleets.

API Server Settings General

You can set the following options for the API server:

API Server Setting	Description
Host	The hostname or IP address you want to bind the API server to. • Defaults to `127.0.0.1` for standalone Edge Nodes and Edge Fleets. • You can override the value with the `CRIBL_API_HOST` environment variable.
Port	API port to listen to. • Defaults to `9420` for standalone Edge Nodes and Edge Fleets. • You can override the value with the `CRIBL_API_PORT` environment variable.

TLS

For information on TLS options, see the documentation for any Source or Destination that supports TLS.

Advanced

You can set the following advanced options for the API server:

API Advanced Server Setting	Description
Retry count	The number of times to retry binding to the API port. Default is `120`.
Retry period	The period between consecutive retries for API port binding, in seconds. Default is `5`.
URL base path	The URL base path from which to serve all assets. Setting a URL base path may be useful when operating behind a proxy server.
Listen on port:	Toggle on (default) to expose the API service to the network on the configured API port. When toggled off, Cribl Edge doesn't listen on the port (`9420` by default).
Local UI access	Toggle on (default) to allow direct browser access to the UI for Edge Nodes.

API Advanced Server Setting	Description
Logout on roles change	If role-based access control is enabled, toggle on to automatically log out users when their assigned Roles change. Default is toggled on.
Auth token TTL	Authentication tokens' valid lifetime, in seconds. Default is `3600` (60 minutes = 1 hour); minimum is `1`.
Session idle time limit	How long to observe no user interaction before invalidating users' session tokens, in seconds. Default is `3600` (60 minutes = 1 hour); minimum is `60`.
Login rate limit	The number of login attempts allowed over the specified unit of time. For example, to limit login attempts to 50 per minute, specify the login rate limit `50/minute`. Valid units of time are `second`, `minute`, `hour`, and `day`. Default is `2/second`.
SSO/SLO callback rate limit	The number of requests to SSO and SLO callback endpoints allowed over the specified unit of time. For example, to limit requests to 10 per minute, specify `10/minute`. Valid units of time are `second`, `minute`, `hour`, and `day`.
HTTP headers	One or more custom HTTP headers to send with every response.
Enable API cache	Toggle on to enable browser caching of frequent API requests. Default is toggled on. Toggling off can slow the the UI response time.

Default TLS Settings

See the System-wide TLS Settings Including Ciphers.

Limits

The **Limits** tab provides access to controls for metrics, storage, metadata, jobs, the Redis cache and connections to it, and CPU settings.

Metrics

Metrics Field	Description
Metrics worker tracking	Define which metrics are not assigned a Node ID dimension, which means they will not be split by Edge Node.
Number of metrics allowed	The total number of metrics series allowed.

Metrics Field	Description
Metrics cardinality limit	The maximum allowed number of unique combinations of metrics names and their dimensions.
Metrics GC period	The metrics garbage collector (GC) runs when the total number of stored metrics exceeds the **Number of metrics allowed**.
Metrics never-drop list	A list of internal metrics that the Cribl Edge Node will always send to the Leader.
Disable field metrics	A list of metrics that you don't want to collect.

Metrics to send from Edge Node	Description
Minimal	Sends metrics for events and bytes in and out. Recommended for high-cardinality deployments. Sent as aggregate totals per Fleet: • `total.in_events` • `total.out_events` • `total.in_bytes` • `total.out_bytes` Excludes metrics that are coming directly from Sources or Destinations.

Basic (default)	Contains all the metrics in the minimal set, and more. Sends metrics required to display all monitoring data available in the Edge UI. Sent as aggregate totals per Source/Destination:
	`total.in_events`, `total.out_events`, `total.in_bytes`, `total.out_bytes`.
	Also, Sent as aggregate totals per Source:
	`health.inputs`
	Sent as aggregate totals per Destination:
	`health.outputs`
	Sent as aggregate totals per Route:
	`route.in_events`, `route.in_bytes`, `route.out_events`, `route.out_bytes`, `route.dropped_events`
	Sent as aggregate totals per Pipeline:
	`pipe.in_events`, `pipe.out_events`, `pipe.dropped_events`
	Sent as aggregate totals per Pack:

Metrics to send from Edge Node	Description
	`pack.in_events`, `pack.out_events`, `pack.err_events`, `pack.dropped_events`
All	Send all metrics with no filtering. All metrics without filtering, including **Basic**, plus Worker/Edge Node Resource Metrics, Persistent Queue Metrics, and Other Internal Metrics.
Custom	Send a customized set of metrics. All metrics that match a JavaScript expression that you define, plus the metrics in the **Minimal** set.

For more information about each of the settings above, go to Controlling Metrics Volume.

Storage

You can configure the following options for storage:

Storage Setting	Description
Sample size limit	Maximum file size, in binary units (`KB`, `MB`), for sample data files. Maximum: `3 MB`. Default: `256 KB`.
Min free disk space	The minimum amount of disk space on the host before various features take measures to prevent disk usage (KB, MB, etc.). Default: `5 GB`. To avoid the "Unable to write to the filesystem, disk space low" error, ensure that the available disk space remains above this threshold by monitoring disk usage and configuring alerts.
Worker Process PQ size limit	Highest accepted value for the **Queue size limit** option used in individual Sources' and Destinations' persistent queues. Default: `1 TB`. Consult Cribl Support before increasing beyond this value.

Metadata

Event metadata sources: List of event metadata sources to enable. No sources are enabled by default.

Jobs and Tasks

Job Limits Setting	Description
Disable jobs/tasks	When enabled, the Edge Nodes won't poll the Leader for jobs/tasks. The job limits settings below will not affect Edge Nodes on version 4.5.0 and newer. Edge Nodes running 4.4.4 and older still use these jobs settings even if jobs/tasks are disabled here.
Concurrent job limit	The total number of jobs that can run concurrently. Defaults to `10`.
Concurrent system job limit	The total number of **system** jobs that can run concurrently. Defaults to `10`. Minimum `1`.
Concurrent scheduled job limit	The total number of **scheduled** jobs that can run concurrently. This limit is set as an offset relative to the **Concurrent job limit**. Defaults to `-2`.



Skipped jobs indicate that a Fleet's **Concurrent job limit** has been reached or exceeded. Increase this limit to reduce the number of skippable jobs. For resource-intensive jobs, this might require deploying more Edge Nodes.

Task Limits Setting	Description
Concurrent task limit	The total number of tasks that a Worker Process can run concurrently. Defaults to `2`. Minimum `1`.
Concurrent system task limit	The number of system tasks that a Worker Process can run concurrently. Defaults to `1`. Minimum `1`.
Task usage percentage limit	Value, between `0` and `1`, representing the percentage of total tasks on a Worker Process that any single job may consume. Defaults to `0.5` (50%).
Task poll timeout	The number of milliseconds that a Worker's task handler will wait to receive a task, before retrying a request for a task. Defaults to `60000` (60 seconds). Minimum `10000` (10 seconds).

Completion Limits

Completion Limits Setting	Description
Artifact reaper period	Interval on which Cribl Edge attempts to reap jobs' stale disk artifacts. Defaults to `30m`.

Completion Limits Setting	Description
Finished job artifacts limit	Maximum number of finished job artifacts to keep on disk. Defaults to `100`. Minimum `0`.
Finished task artifacts limit	Maximum number of finished task artifacts to keep on disk, per job, on each Edge Node. Defaults to `500`. Minimum `0`.

Task Manifest and Buffering Limits

Task Manifest and Buffering Setting	Description
Manifest flush period	The rate (in milliseconds) at which to refresh the task manifest of a job. Defaults to `100` ms. Minimum `100`, maximum `10000`.
Manifest buffer size limit	The maximum number of tasks that the task manifest can hold in memory before flushing to disk. Defaults to `1000`. Minimum `100`, maximum `10000`.
Manifest reader buffer size	The number of bytes that the task manifest reader should pull from disk. Defaults to `4kb`.
Job dispatching	The method by which tasks are assigned to Worker Processes. Defaults to `Least In-Flight Tasks`, to optimize available capacity. `Round Robin` is also available.
Job timeout	Maximum time a job is allowed to run. Defaults to `0`, for unlimited time. Units are seconds if not specified. Sample entries: `30`, `45s`, `15m`.
Task heartbeat period	The heartbeat period (in seconds) for tasks to report back to the Leader/API. Defaults to `60` seconds. Minimum `60`.

Redis Cache

Redis Cache Setting	Description
Key TTL in seconds	Maximum time to live of a key in the cache (seconds). `0` indicates no limit. Defaults to `10 minutes`.
Max # of keys	Maximum number of keys to retain in the cache. `0` indicates no limit. Defaults to `0`.

Redis Cache Setting	Description
Max cache size (bytes)	Maximum number of bytes to retain in the cache. `0` indicates no limit. Defaults to `0`.
Service period (seconds)	Frequency of cache limit enforcement. Defaults to every `30 seconds`.
Server assisted	Default is toggled off. When toggled on, the following control appears.
Client tracking mechanism:	Mechanism for invalidation message delivery. In default mode, the server remembers which keys a client has requested and only sends invalidations for those, using more Redis server memory. In broadcast mode, it sends all invalidations, requiring more processing by Cribl Edge.

Connections

Redis Connections Setting	Description
Reuse Redis connections	Toggle on if you want Cribl Edge to try to reuse Redis connections when multiple Redis Functions (or references to them) are present. When enabled, displays the following additional control: • **Connection limit**: The maximum number of identical connections allowed before Cribl Edge tries to reuse connections. Defaults to `0`, meaning unlimited connections are allowed (equivalent to leaving **Reuse Redis connections** toggled off). Setting a non-zero integer value forces Cribl Edge to try to reuse connections for each individual Worker Process (**not** to reuse connections **among** Worker Processes).

To understand why and when to employ these controls, see Reusing Redis Connections.

Other

Other Setting	Description
CPU profile TTL	The time-to-live for collected CPU profiles.
Default managed node heartbeat period	How many seconds a managed Node will wait to send back a heartbeat to the Cribl control plane.

Other Setting	Description
Config bundle download timeout	How many seconds a Cribl Stream Worker will wait for a successful Leader connection before canceling a download of a new configuration bundle. This timeout helps prevent Workers from hanging indefinitely when there are network issues or other delays during the download process. A `0` value means wait indefinitely, which could cause Workers to hang.
Minimum reconnect interval	The minimum time to wait before attempting to reconnect to the Leader after being disconnected or when the Leader is not available. The interval is doubled with each failed reconnect attempt until the maximum limit is reached.
Maximum reconnect interval	The maximum time to wait before attempting to reconnect to the Leader after being disconnected or when the Leader is not available.
Random reconnect interval	Upper limit on the random time added to each reconnect interval. Set this to a value other than zero to randomly spread out reconnection attempts. This can prevent the Leader being overwhelmed by reconnection attempts by a large number of Edge Nodes.
Use Windows tools to collect process info	Enable PowerShell to collect process information instead of using the native API. This legacy setting will be removed in a future release. We highly recommend keeping this option disabled and using the newer native capabilities which are faster and more reliable.



When using PowerShell to collect process information, the environment variables displayed in the Explore > Processes tab might be inaccurate. They will reflect the environment of the PowerShell process itself, not the actual environment of the process being viewed.

Proxy Settings

Use proxy env vars: Honors the `HTTP_PROXY/HTTPS_PROXY` environment variables. Defaults to toggled on.

Cribl prioritizes environment variables for proxy settings in this order: Process, User, and System.

If your Cribl service is managed by a service manager other than systemd (such as upstart or init), the `Use proxy env vars` toggle might not behave as expected because Cribl might prioritize environment variables set by the service manager instead of using the proxy settings you intended.

Sockets

Directory: Holds sockets for inter-process communication (IPC), such as communications between a load-balancing process and a Worker Process. Defaults to `/tmp` (your system's temp directory).

Shutdown Settings

Drain timeout (sec): Determines how long a Cribl server will wait for writes to complete before the server shuts down on individual Worker Processes. If you notice that Workers are under-ingesting available data upon shutdown or restart, increase the `10`-second default. Acceptable range of values: minimum `1` second, maximum `600` seconds (10 minutes).

Worker Processes

Worker Processes Setting	Description
Process count	**This setting only applies to the** Kubernetes Logs Source. Indicates the number of Worker Processes to spawn. Positive numbers specify an absolute number of Workers. Negative numbers specify a number of Workers relative to the number of CPUs in the system, for example: {`<number of CPUs available>` minus `<this setting>`}. The default in Cribl Edge is 1. • You can use **Process count** to adjust the number of Worker Processes **only for the Kubernetes Logs Source**. This is necessary when running the Kubernetes Logs Source with load balancing enabled. Load balancing increases performance when you have a higher event ingest volume than one Worker Process can handle. • Cribl Edge will provision at least the **Minimum process count** set below, but no more than the number of CPUs the host has available.
Minimum process count	Indicates the minimum number of Worker Processes to spawn. Overrides the lowest Process count result. Cribl Edge interprets a 0 entry as "default," which here yields 1 Process.
Memory (MB)	Amount of heap memory available to each Worker Process, in MB. The OS allocates heap memory dynamically as the Process requests it up to the amount dictated by this setting. The Memory (MB) setting does not govern external (ext) memory.
Enable heap snapshots	Default is toggled off. Toggle on for Cribl Edge to automatically create memory snapshots for Worker Processes when they approach or exceed memory limits. Only the two most recent heap snapshots are retained. Older snapshots are automatically deleted. This behavior cannot be modified.

The **Enable heap snapshots** is available for hybrid or on-prem deployments only. This setting should only be enabled if recommended by support or if you are experiencing out-of-memory issues. Be aware that enabling this setting may impact system performance and storage usage.

Non-configurable Settings

These settings can't be configured in Cribl Edge, and instead have default values.

Max connections at startup: Maximum number of connections accepted at Worker Process startup. Defaults to `1`.

8.1.4. Create And Manage Fleets And Subfleets

Learn how to create Fleets and Subfleets in Cribl Edge.

Fleets and their corresponding Subfleets allow you to author and manage configuration settings for a particular set of Edge Nodes, so you can flexibly group Edge Nodes into logical Fleets and Subfleets. These Fleets and Subfleets can share and reuse configurations.

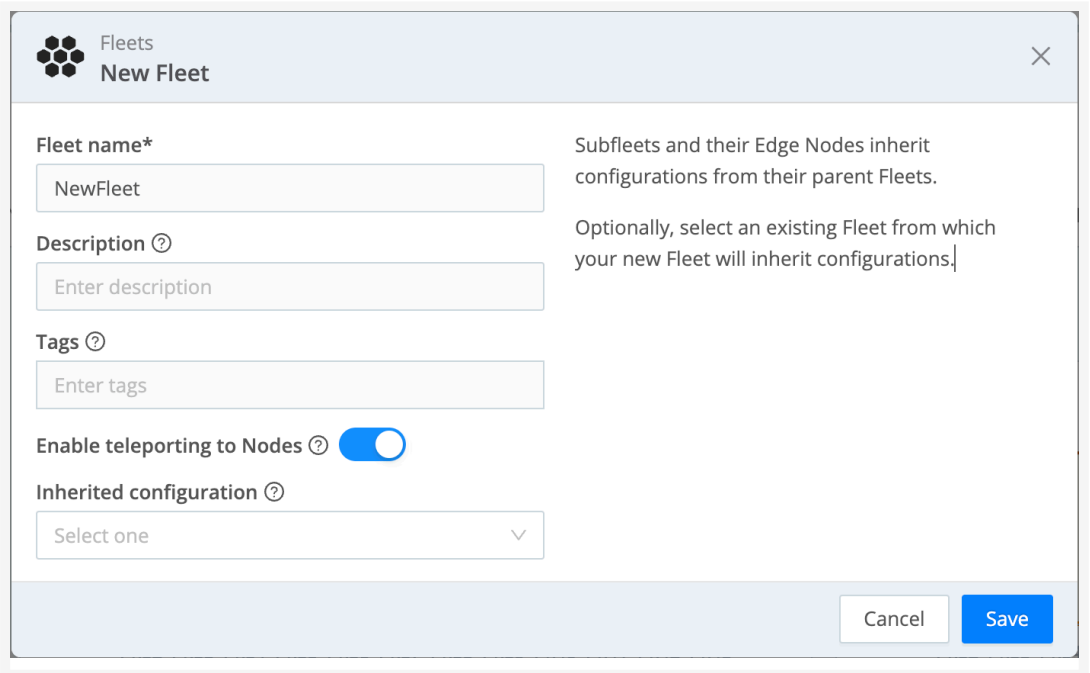
You can assign Fleet-level permissions to users. For details, see Members and Permissions.

Create a Fleet

Required Permission: Cribl Edge Admin

To create a new Fleet:

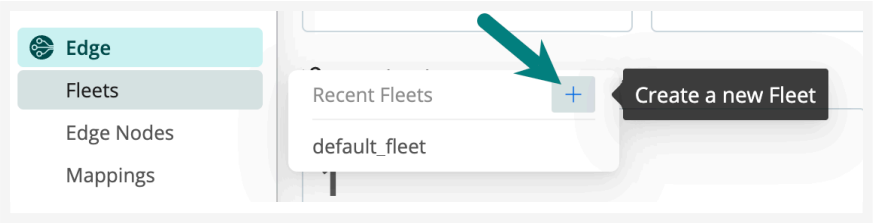
1. In the sidebar, click **Fleets**.
2. Click **New Fleet**.
3. Enter a descriptive **Fleet name**, and optionally, a relevant **Description** and Tags.



Add a new Fleet

To enable teleporting to the Edge Nodes via the Leader's UI, toggle **Enable teleporting to Nodes** to **Yes**, then click **Save**.

You can also create a new Fleet by selecting **Products** on the top bar, hovering over **Fleets** in the sidebar, and selecting the + button.



Creating a New Fleet from **Products** Sidebar

Cribl.Cloud Fleet Limits

In Cribl.Cloud, you can create up to a combined total of 250 Fleets and Subfleets - for example, 50 Fleets + 200 Subfleets.

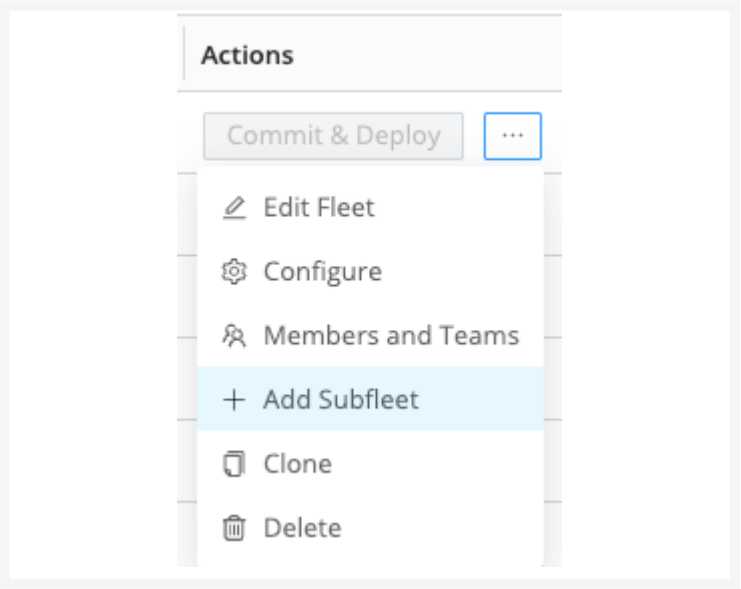
You can contact Cribl Support for an increase if you have a use case that requires more than 250 Fleets/Subfleets.

Add a Subfleet

Cribl recommends no more than two levels of Subfleets nested below a parent Fleet, but if your setup requires, you can create up to four levels of Subfleets. For each Subfleet, you can choose which parent Fleet it will inherit configurations from. For more information about inheriting configurations, see Fleet Inheritance.

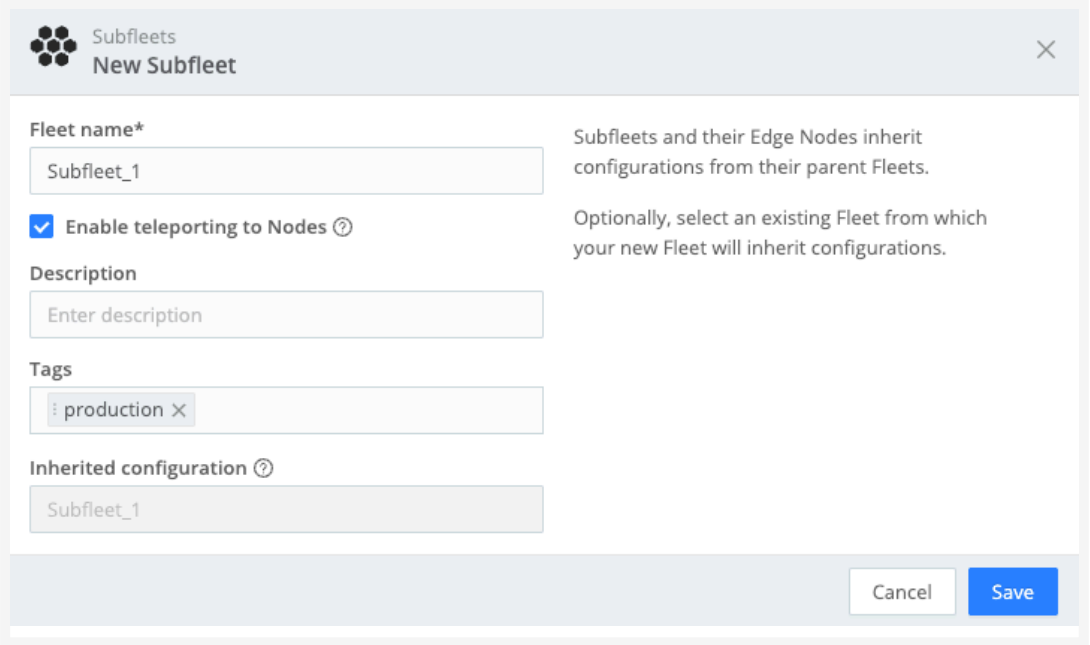
To add a Subfleet:

1. In the row of your desired Fleet, select the **Options** (···) menu, then select **Add Subfleet**.



Menu showing the Add Subfleet option from Options

2. Enter a descriptive **Fleet name** and optional **Description** and Tags.
3. Select an existing Fleet from the **Inherited configuration** drop-down. The Subfleet will inherit its configuration from the selected Fleet.
4. Click **Save**.

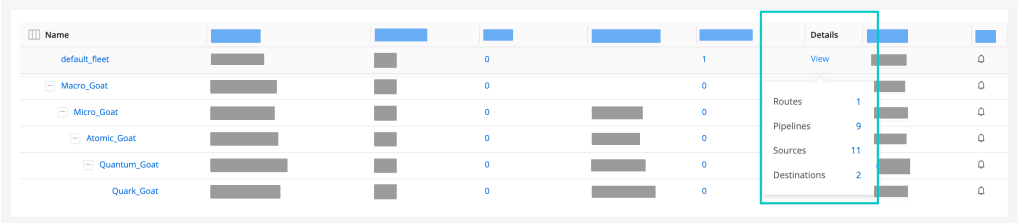


Add a Subfleet

Similarly, you can create a Subfleet in a selected Subfleet. Each Subfleet will inherit the configurations of every parent Subfleet before it.

View Subfleet Details and Options

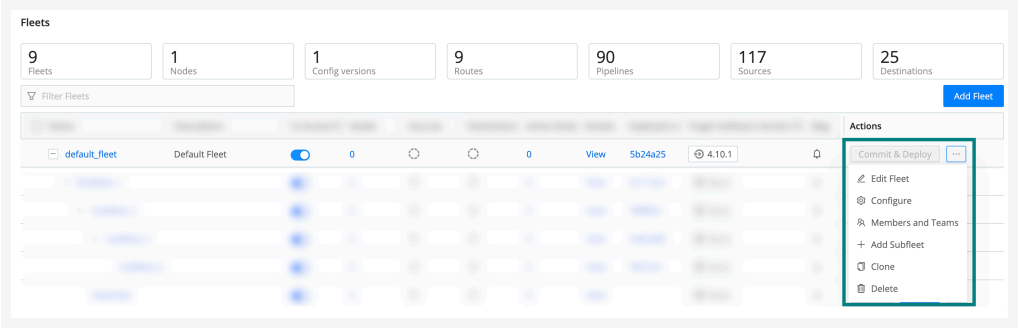
To display the number of Routes, Pipelines, Sources, and Destinations in a Fleet, click **View** in the **Details** column of a Fleet or Subfleet.



View Subfleet details

On Cribl.Cloud, you'll also see a **Group Type** column, showing whether the deployment is **Hybrid** or **Cloud**.

To display the **Edit Fleet**, Configure, **Members and Teams**, **Add Subfleet**, **Clone**, and **Delete** options, select the **Options** (⋯) menu.

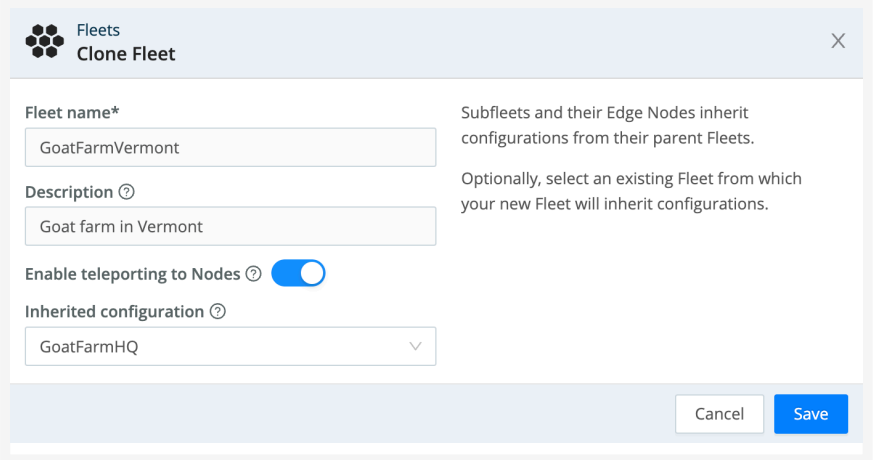


A menu flyout listing the Fleet options

Clone and Delete a Subfleet

Cloning a Fleet copies the Fleet's settings, while cloning a Subfleet inherits the parent's configurations.

To clone a Subfleet, click its **Options** (···) menu and select **Clone**.



Clone a Subfleet

To delete a Fleet or Subfleet, click **Delete** in the **Options (•••)** menu. You will be prompted to confirm your decision to delete the Fleet/Subfleet, click **Yes** to confirm.

You can also commit and deploy changes to Fleets and Subfleets on this page. For details, see Committing and Deploying Fleet Configurations.

Configuring multiple Fleets is available with an Enterprise plan. For details, go to the Edge - Cloud Ingest guide.

Edge Node GUID

When you install and first run the software, a GUID is generated and stored in a `.dat` file located in `CRIBL_HOME/local/cribl/auth/`.

For example:

```
# cat CRIBL_HOME/local/cribl/auth/676f6174733432.dat

{"it":1570724418,"phf":0,"guid":"48f7b21a-0c03-45e0-a699-01e0b7a1e061"}
```

When deploying Cribl Edge as part of a host image or VM, be sure to remove this file, so that you don't end up with duplicate GUIDs. The file will be regenerated on the next run.

8.1.5. Configure Fleets And Deploy Changes

Learn how to configure Fleets and Subfleets, then commit and deploy the changes.

Cribl Edge provides you with the ability to configure Fleet and Subfleets. At the Fleet and Subfleet level, you can add, update, and remove:

- Sources
- Destinations
- Routes
- Pipelines
- Packs

To access these configuration options:

- Click the **Options (•••)** menu on a Fleet/Subfleet, then click **Configure**. This takes you to **Collect**, where you can use QuickConnect to quickly set up data collection Sources and hook them up to Destinations.
- Click a Fleet **Name**, then click **Collect** or use the **More** drop-down.

When you make changes to a Fleet, you must commit and deploy the changes in order for the Edge Nodes to receive the updated configs.

Example Configuration Deployment Workflow

Here's a typical workflow for deploying Edge configurations:

1. Make configuration changes on a Fleet.
2. Save your changes.
3. Commit (and optionally push).
4. Locate your desired Fleet and click **Deploy**.

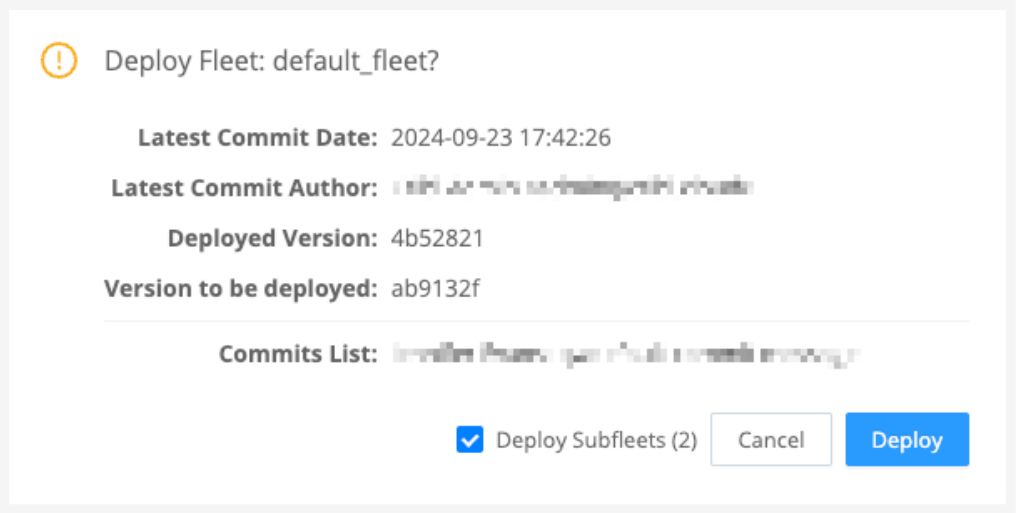
Once you save and commit your configuration changes, you can deploy the updated configs to the Edge Nodes. You deploy new configurations at the Fleet level.

Deploy Subfleets

To make new configurations visible and available to Subfleets, you must first **Commit** the changes within parent Fleets' configurations.

When you commit changes to a parent Fleet, you have the option to **Deploy Subfleets** at the same time.

When you **Deploy Subfleets**, the Subfleets will receive the new inherited configuration. For guidelines on deploying to a large number of Edge Nodes, see Leader Scalability.



Deploy Subfleets

Edge Nodes that belong to the Fleet/Subfleet will start **pulling** updated configurations on their next check-in with the Leader.

Can't log in to the Edge Node as Admin User?

When an Edge Node pulls its first configs, the admin password will be randomized, unless specifically changed. This means that users won't be able to log in on the Edge Node with default credentials. For details, see User Authentication.

Configuration Files

On the Leader, a Fleet's configuration resides in this directory:

`$CRIBL_HOME/groups/<FleetName>/local/edge/`

Where each Fleet gets its own directory that contains a collection of `.yml` config files.

The managed Edge Node pulls configs and extracts them under:

`$CRIBL_HOME/local/edge/.`

Some configuration changes will require restarts, while many others will require only reloads. For details, see Configurations and Restart. Upon restart, individual Edge Nodes might temporarily disappear from the Leader's **Nodes** tab, before reappearing.

Notable Environment Variables for a Fleet

- `CRIBL_DIST_LEADER_URL` - URL of the Leader Node. Format: `<tls|tcp>://<authToken>@host:port?group=default_fleet&tag=tag1&tag=tag2&tls.<tls-settings below>`.
  Example: `CRIBL_DIST_LEADER_URL=tls://<authToken>@leader:4200`
  Parameters:
  - `group` - The preferred Fleet assignment.
  - `resiliency` - The preferred Leader failover mode.
  - `volume` - The location of the NFS directory to support Leader failover.
  - `tag` - A list of tags that you can use to assign the Edge Node to a Fleet.
  - `tls.privKeyPath` - Private Key Path.
  - `tls.passphrase` - Key Passphrase.
  - `tls.caPath` - CA Certificate Path.
  - `tls.certPath` - Certificate Path.
  - `tls.rejectUnauthorized` - Validate Client Certs. Boolean, defaults to `false`.
  - `tls.requestCert` - Authenticate Client (mutual auth). Boolean, defaults to `false`.
  - `tls.commonNameRegex` - Regex matching peer certificate > subject > common names allowed to connect. Used only if `tls.requestCert` is set to `true`.
- `CRIBL_DIST_MODE` - Set to one of: `managed-edge` (managed Node), `leader` (Leader instance), or `edge` (single instance).
- `CRIBL_HOME` - Auto setup on startup. Defaults to parent of `bin` directory.
- `CRIBL_CONF_DIR` - Auto setup on startup. Defaults to parent of `bin` directory.
- `CRIBL_DIST_LEADER_BUNDLE_URL` - AWS S3 bucket (format: `s3://${bucket}`) for remote bundle storage.
- `CRIBL_TMP_DIR` - Defines the root of a temporary directory.
  Sources use this directory to stage downloaded Parquet data. The format of derived temporary directories is: `$CRIBL_TMP_DIR || os.tmpdir())/cribl_temp[/<componentPath>/<PID>]` ... where: `<componentPath>` = `<inputId>`; `<PID>` = Worker Process ID; and `os.tmpdir()` defaults to `/tmp` on Linux. Override that default using this `CRIBL_TMP_DIR` environment variable.
- `CRIBL_VOLUME_DIR` - Sets a directory that persists modified data between different containers or ephemeral instances.

- `CRIBL_DIST_WORKER_PROXY` - Communicate to the Leader Node via a SOCKS proxy. See `CRIBL_DIST_WORKER_PROXY` for more details.

8.1.6. Recover A Deleted Fleet

Learn how to recover deleted Fleets in Cribl Edge.

If you delete a Fleet, you can no longer access its **Version Control** menu. As a result, you also lose the option to directly recover the Fleet.

However, you can still recover a deleted Fleet using the command line. To revert the deletion commit:

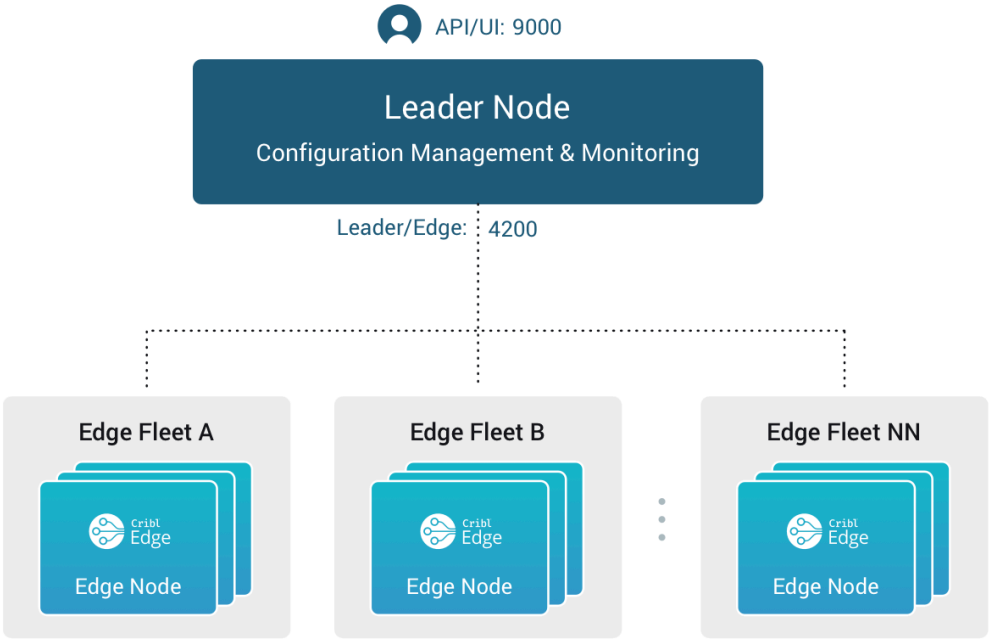
1. Stop the Cribl server.
2. Using a command line interface, look at the version control history for the Leader Node and identify the commit which deleted the Fleet.
3. Revert to the last commit before the Fleet was deleted.
4. Restart the Cribl server.

8.2. Edge Nodes

Learn how Edge Nodes and Leader Nodes work together

The Leader Node has two primary roles:

1. Serves as a central location for Edge Nodes' operational metrics. The Leader ships with a monitoring console that has a number of dashboards, covering almost every operational aspect of the deployment.
2. Serves as a central location for authoring, validating, deploying, and synchronizing configurations across Fleets.



Leader Node/Edge Nodes relationship

Network Port Requirements (Defaults)

- UI access to Leader Node: TCP 9000.
- Edge Node to Leader Node: TCP 4200. Used for Heartbeat/Metrics/Leader requests/notifications to clients (such as live captures, teleporting, status updates, and config bundle notifications).
- Edge Node to Leader Node: HTTPS 4200. Used for config bundle downloads.
- Edge Node bootstrapping/update: HTTPS 443.

Leader/Edge Node Communication

Edge Nodes will send a heartbeat to the Leader every 60 seconds. This heartbeat includes information about the Nodes and a set of current system metrics. The heartbeat payload includes facts - such as hostname, IP address, GUID, tags, environment variables, and current software/configuration version - that the Leader tracks with the connection.

When an Edge Node checks in with the Leader:

- It sends a heartbeat to Leader with “facts” about itself.
- The Leader uses Edge Node’s facts and Mapping Rules to map it to a Fleet.
- The Edge Node pulls its Fleet’s updated configuration bundle, if necessary.

Edge Nodes that don’t maintain their connection to the Leader will eventually disappear from the Edge Nodes page.



The Leader is unaware of the platforms on which Edge Nodes within a Fleet run. So the ConfigHelper omits platform-specific limitations. Therefore, when you manage Edge Nodes on heterogeneous platforms, create OS-specific Fleets (for example, a Windows Fleet, a macOS Fleet) and mapping. See Design Fleet Hierarchy.

Leader Disconnection and Telemetry

Cribl Edge is built to keep your data flowing smoothly, even if it temporarily loses connection with the Leader Node. This means you won’t experience disruptions during network hiccups or if the Leader is briefly unavailable. Cribl Edge periodically sends telemetry to verify its license with Cribl’s servers. This is different from its connection to the Leader. If the telemetry fails for an excessive time, processing might be interrupted.



This telemetry is disabled for Cribl.Cloud-managed Edge Nodes.

Cribl Edge Nodes that are disconnected from the Leader will resume operation after an Edge restart, using their last settings and license. In most cases, an Edge Node gets its license from the Leader, so if it restarts without a connection, it will use a temporary free license and continue its check-ins. Just keep in mind that while the Edge Node is disconnected, you won’t see real-time updates in the main monitoring dashboard, and it won’t receive any new configuration changes.

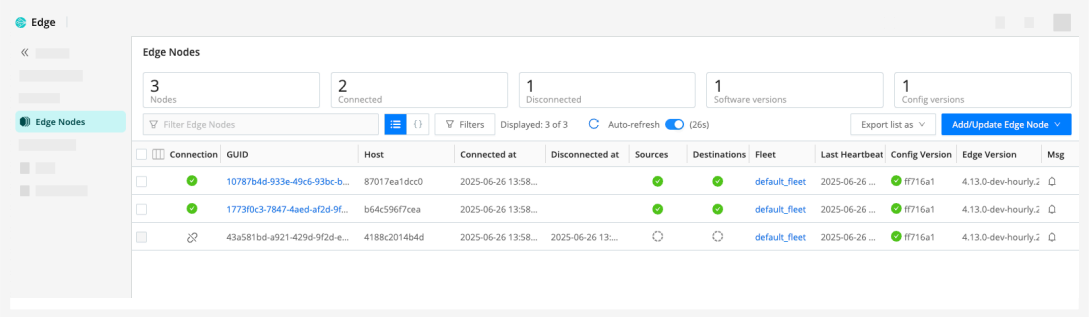
8.2.1. Manage Edge Nodes

How to manage Edge Nodes in a Fleet and perform tasks like teleporting and exporting a list of Nodes.

View Edge Node Status Information

The **Edge Nodes** page provides status information for each Edge Node in the selected Fleet, as well as a UI for adding and updating Nodes.

In Cribl Edge, select **Edge Nodes** from the sidebar.



Edge Nodes page showing a list of Nodes

Monitor Edge Node Connectivity

You can track which Edge Nodes are connected to the Leader Node in the **Edge Nodes** page. Disconnected Edge Nodes show an icon in the **Connection** column and will have information about the time the Leader last received their heartbeat.

By default, Cribl Edge will retain information about disconnected Nodes between restarts. To change this behavior on-prem, toggle **Disable Node persistence** on in **Global Settings** > **System** > **General Settings** > **Limits** > **Other**. This will also affect Cribl Stream Workers.

Disable Node persistence is a deprecated setting that will be removed in a future version. Toggling **Disable Node persistence** will restart the Leader Node.

You can configure how long the Leader should keep track of disconnected Nodes by configuring the **Time to keep disconnected Nodes** Fleet setting (**General Settings** > **Fleet Configuration**). By default the Leader retains information about disconnected Nodes for one day. You can set it to any value between 0 (which means Nodes are removed within a few minutes) and 90 days.

For Fleets with Edge Nodes that frequently restart with a new GUID, such as those in a Kubernetes environment, Cribl recommends setting the **Time to keep disconnected Nodes** to 0.



The **Time to keep disconnected Nodes** setting is not inherited by Subfleets.

Monitor Connectivity of Older Edge Nodes

A Leader Node that is running version 4.13 or newer can monitor Edge Nodes that are using an older version. However, Cribl always recommends upgrading your Edge Nodes to the newest available version.

Monitor Source and Destination Health on Edge Nodes

You can monitor and track the health of Sources and Destinations associated with an Edge Node on the **Edge Nodes** page. This is useful when you have a large number of Nodes, and want to quickly identify and troubleshoot issues with Sources and Destinations.

To open the Node Info drawer and view specific monitoring information, select a health icon. For more information, see Track Data Activity for an Edge Node.

Filter Nodes in the List

You can filter the Edge Nodes that appear in the list using the **Filters** option. For details, see Filter Edge Nodes.

Find Edge Nodes Running in a Container

Edge Nodes running in a container report additional metadata about the container to the Leader. This helps you quickly identify which of your Edge Nodes are running in a container from the Leader UI and map them to the appropriate Fleets.

When Cribl Edge is running in a container, you will see the `hostOs` property in events collected by the System State Source. This object will contain properties about the container when they're available, such as:

- `arch`
- `cpu_count`
- `cpu_speed_mhz`
- `cpu_type`
- `interfaces`
- `memory`
- `platform`

- `release`
- `hostname`
- `timezone`

From the **Mappings** menu, you will also see the `hostOs` property in the **Edge Node info** panel. This object will include the `id`, `addresses`, and `version` of the container when available.

`interfaces` is a set of key-value pairs that contain an array of CIDR values.

Export a List of Edge Nodes

To export the list of Edge Nodes, click the **Export list as** drop-down. Select `JSON` or `CSV` as the export file format. If you filter the list before exporting, the exported list will maintain that filter. You can export the entire list by removing all filters.

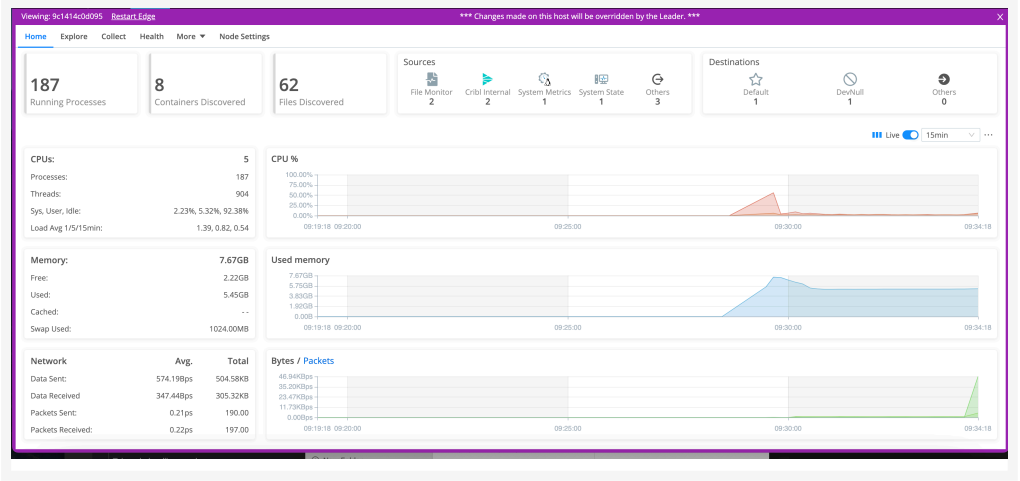
Teleport into an Edge Node

Click the **Edge Node GUID** link to teleport from the Leader into the Edge Node. Here, you can explore the metrics and log data that the Node has discovered automatically, and can manually discover and explore other data of interest.

You can use the discovered data to perform root-cause analysis, to troubleshoot, and to restart the host. If you want to monitor and troubleshoot the Node data without teleporting, check out the Node Info Panel.

The page displays metadata for the Node, and below it, graphs of system activity. A magenta border indicates you are remotely viewing a host, and identifies the host's name.

Click **Restart Edge** to restart the Node. To return to the **Edge Nodes** page, click the **X** close button on the upper right.



Teleporting into an Edge Node

Changes that you make on an Edge Node will not propagate to the Leader. Also, the Leader will override any changes that you make directly on a Node.

Add Tags to Edge Nodes in Teleport View

You can add tags to an Edge Node you've teleported into. Tags help the Leader map an Edge Node to a Fleet. Note that changing the tags on a Node can cause the Leader to map the Node into a new Fleet.

1. Teleport into an Edge Node.
2. Navigate to **Node Settings** then **Distributed Settings**.
3. Enter tags for this Node and select **Save**.

If your Edge Nodes are managed by Cribl.Cloud, you'll need to accept the confirmation modal. Keep in mind that changing the tags on an Edge Node might cause it to be assigned to a different Fleet by the Leader.

You can't change tags for Edge Nodes set up using the `CRIBL_DIST_MASTER_URL` environment variable (instead of the `instance.yml` file). This is because these types of Edge Nodes are typically temporary, so any tag changes would not be saved when the Edge Node restarts. If you need tags that persist, configure your Edge Nodes using the `instance.yml` file.

Troubleshooting the Edge Node List

If you see unexpected results in the **Edge Nodes** list, keep in mind that:

- Edge Nodes that don't maintain their connection to the Leader will eventually disappear from the Edge Nodes page.
- For a newly created Fleet, the **Config Version** column can show an indefinitely spinning progress spinner for that Fleet. This happens because the Edge Nodes are polling for a config bundle that hasn't been deployed. To resolve this, click the **Deploy** option to force a deploy.
- For details on collocating multiple Cribl products, see Cribl Edge and Cribl Stream on the Same Host.
- For details on overriding default ports, see Overriding Default Ports.

8.2.2. Add And Update Edge Nodes

Learn how to add and update Edge Nodes in Cribl Edge

Cribl Edge provides you with the ability to add and update Edge Nodes from the UI. To access this function, select **Edge Nodes** in the sidebar. On the **Edge Nodes** page, select **Add/Update Edge Node**.



Options to add or update an Edge Node

Here, you can update an existing Node with new configurations or add a new Node using the bootstrap script.

Add a New Edge Node with a Bootstrap Script

Using the Cribl Edge UI, you can add Edge Nodes to your existing deployment. To do this, change the environment-specific fields, then copy and paste the resulting bootstrap script into a CLI. The script updates in real-time as you change the fields. The result is a new Edge Node configured with the options you chose.

You can also copy the bootstrap script and modify it instead of manually editing the fields in the UI, as well as concatenate multiple scripts for different use cases.

Host Communication Prerequisites

To enable the Leader to manage the Edge Nodes, you must enable ongoing outbound communication to the Leader's port 4200 on all Edge Nodes hosts.

While the bootstrap script runs, firewalls on each Nodes's host must also allow outbound communication on the following ports:

- Port 443 to `https://cdn.cribl.io`, when **Install package location** is set to `Cribl CDN`.
- Port 443 to a Cribl.Cloud Leader.
- Port 9000 to an on-prem Leader.

Add an Edge Node

1. From the Cribl Edge sidebar, select **Edge Nodes**.
2. On the **Edge Nodes** page, select **Add/Update Edge Node**.
3. Select your OS, then select **Add**.

Linux Windows macOS

1. Fill and modify the deployment option fields:
    - The **Install package location** defaults to `Cribl CDN`. If desired, change this to `Download URL`.
    - As needed, correct the target **Fleet**, as well as the **Leader or Outpost hostname/IP** (URI).
    - As needed, correct the **User** and **User Group** to run Cribl as.
    - As needed, correct the **Installation directory**.
2. Optionally, add **Tags** that you can use for filtering and grouping in Cribl Edge. Use a tab or hard return between (arbitrary) tag names.
3. Copy the resulting script to your clipboard.

Finally, to add the Edge Node, paste the script onto the command line of your Edge Node and execute it.

If your script successfully added the Edge Node, you will see it in the **Edge Nodes** list.

Update an Edge Node with a Bootstrap Script

You can use bootstrap scripts to update the configuration of your Edge Node.

Updating adjusts the Leader connection details for the Node, retaining its version. To upgrade the Edge Node - that is, install a newer version of the software - see Upgrading.

Select Update Script Type

When you update a Linux or macOS Node, you must select a **Script type** from the drop-down. Choose between environment variable or CLI to add/update an Edge Node.

With **Environment variable** selected:

1. Hover over the **Script** field and select the copy icon.

2. To allow Edge to update or add Edge Nodes, configure the environment variable with a restart command.

- You can add the restart command to the environment variable in the Linux systemd service unit file, so you don't have to insert the environment variable each time you restart the server:

    1. In the CLI, enter `sudo systemctl edit cribl-edge` on the Edge Node.
    2. Add `ENVIRONMENT="CRIBL_DIST_LEADER_URL=tcp://criblmaster@masterHostname:4200"` to the opened file and save it.
    3. Reload the service with `sudo systemctl restart cribl-edge`.

- Or, you can use use `./cribl restart`, but you will have to insert the environment variable each time you restart the server:

```
CRIBL_DIST_LEADER_URL=tcp://criblmaster@masterHostname:4200 ./cribl restart
```

Copy

With **CLI** selected:

1. Hover over the **Script** field and select the copy icon.
2. Paste the copied script into a CLI and run it.

Edge Node Updates Using Fleet Version Settings

The script you generate using the **Add/Update Edge Node** window ensures accurate updates by using the specified Target Software Version for your Fleet, enhancing the reliability and consistency of updates across all supported environments: Docker, Kubernetes, Linux, macOS, and Windows.

For Docker and Kubernetes:

- If you don't specify a target version, the command uses the Leader version's image and version flag respectively, by default.
- If you specify a target version, the command parses the Fleet version and maps it to the appropriate CDN URL or version flag.

For Linux and macOS environments:

- The existing script behavior from the Leader manages the updates unless you specify a target version, in which case the script uses that version.

For Windows:

- The default behavior uses the Leader Version from the CDN, while specifying a target version directs the command to parse the Fleet version and map it for MSI download via the CDN.

This mechanism ensures a more consistent and reliable deployment process, maintaining version control and improving update accuracy across all environments.

8.2.3. Filter Edge Nodes

Learn how to filter large lists of Edge Nodes

When your Cribl Edge deployment has a large number of Nodes, you may want to filter your Node list down to a manageable subset of Nodes. This can help you monitor your Edge Nodes and troubleshoot any issues.

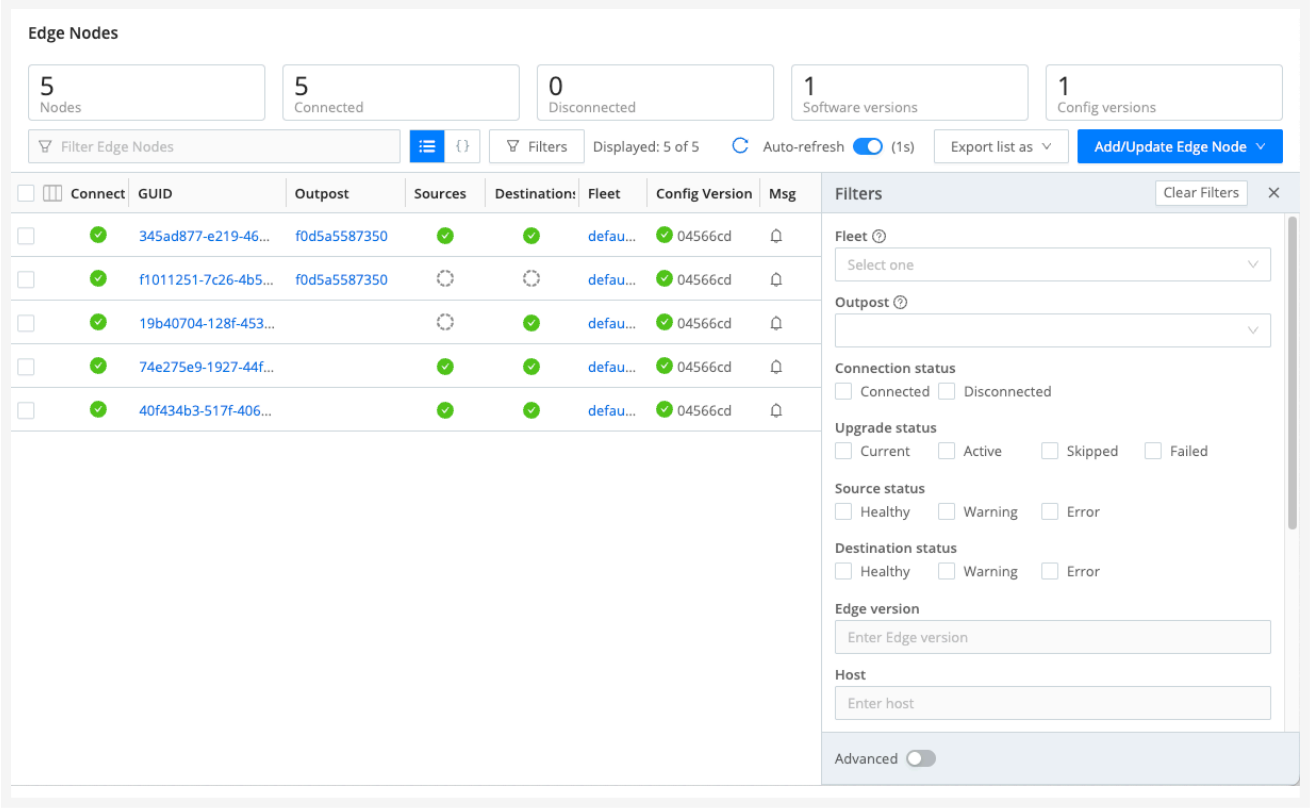
For example, you may want to view all of the Edge Nodes that are experiencing upgrade issues. Or, you may need to update the configuration of Nodes that are running a certain operating system version, but you don't know which Fleets they belong to.

The **Filters** option allows you to apply filter criteria to your Node list, so you can quickly find Nodes by OS Platform, IP address, install type, version and more.

Access Node Filters

In Cribl Edge, you can filter Edge Nodes in two places:

- In a specific Fleet in the Edge Nodes list. Navigate to **Fleets**, then select a Fleet.
- On the list of all Edge Nodes. Navigate to **Edge Nodes** from the sidebar.



The Filters option

Filter Modes

You can choose between the default form-based mode and **Advanced** mode when you create a Node filter. Form-based mode contains the preconfigured fields and checkboxes you see when you select **Filters**.

For a table of the options and their definitions, see Form-based Filter Options.

Advanced mode is a JavaScript input field where you can create your own filter, referencing the JSON objects that are associated with an Edge Node.

You can edit the JavaScript by typing in the **Expression** field, but doing so disables the form-based filters unless you undo the changes. To revert back to the form-based filter, select **Undo changes**.



Some filter combinations will never return any results - for example, **OS platform** set to Linux and **Install type** set to MSI, since MSI is only used for Windows installations.

Create an Edge Node Filter

The **Filters** option lets you group a list of Nodes by criteria such as OS platform, IP address, install type, version, and more.

To create an Edge Node filter:

1. Navigate to a list of Edge Nodes, either from the **Nodes** tab in a Fleet or from the **Edge Nodes** sidebar.
2. Select **Filters** to open the Filters panel.
3. Create a filter by selecting from the Node Filter Options, or toggle **Advanced** on to write your own filter using JavaScript.
4. Once you have selected filter criteria, you can use the **Filter Edge Nodes** field to further narrow down the list of visible Nodes.
5. Select **Clear filters** to undo your selections in the Filters panel. This does not clear the **Filter Edge Nodes** field.

Share a Filter

To share a filter with another Cribl user:

1. Select the **Filters** button and create a filter.
2. Select the link icon to copy the filters to your clipboard, or copy the browser URL.
3. Now you can share the link with another user.

Filters are Workspace-specific, so they will only work within the Workspace where you created them.

Form-based Filter Options

You can choose from a set of preconfigured, form-based filter options, and each has an associated JavaScript property. You'll see the JavaScript property in the **Expressions** field when you toggle **Advanced** on.

Field	Description	JavaScript property
Fleet	Find Nodes based on their Fleet assignment. Visible when filtering from the **Edge Nodes** page.	`group=='<fleet>'`
Outpost	Which Outpost the Edge Node is connected to.	`info.outpost.host.toLowerCase().includes(<outpost-host>)`
Connection status	Whether the Edge Node is currently connected to the Leader.	`(disconnected === <true|false>)`
Upgrade status	The upgrade status of the Edge Node. • **Current**: When the Fleet has a Target Version set, this filters Nodes that are running the same version as the Fleet or a newer version. • **Active**: Nodes that are actively upgrading.	`nodeUpgradeStatus.state`

Field	Description	JavaScript property
	• **Skipped**: Nodes with a skipped upgrade. • **Failed**: Nodes that failed to upgrade.	
Source status	The health status for each Source associated with the Nodes.	`(lastMetrics['health.inputs'])`
Destination status	The health status for each Destination associated with the Nodes.	`(lastMetrics['health.outputs'])`
Edge version	Enter a semantic version of Cribl Edge. For example, 4.1.0.	`info.cribl.version.toLowerCase()`
Host	The host of the Edge Node. You can type in any part of the host string.	`info.hostname.toLowerCase().includes(<host>)`
OS platform	The operating system platform that the Edge Node is running on.	`(info.platform === '<os-platform>')`
OS version	The version of the operating system the Edge Node is running on. For example, Windows 10.	`info.release.toLowerCase().includes('<OS version>')`
Architecture	The processor architecture for the Edge Node.	`(info.architecture === '<architecture>')`

Field	Description	JavaScript property
IP address	The IP address of the Edge Node. Entering a partial value returns all Nodes with matching values. For example, entering `17` will return all Nodes that have `17` somewhere in the address.	`info.conn_ip.toLowerCase().includes('<ip-here>')`
Install type	The installation type for the Node. Select RPM, DEB or Container. Useful for determining which Nodes can't be upgraded by the Leader.	`(info.env.CRIBL_INSTALL_TYPE === '<type>')`

Advanced Filter Options

You can use JavaScript expressions in the Advanced filter to surface more specific, granular search results. You can use any Edge Node properties and types in the **Expression** field.

The JavaScript expression should be in this format, in hierarchical order from parent to child:

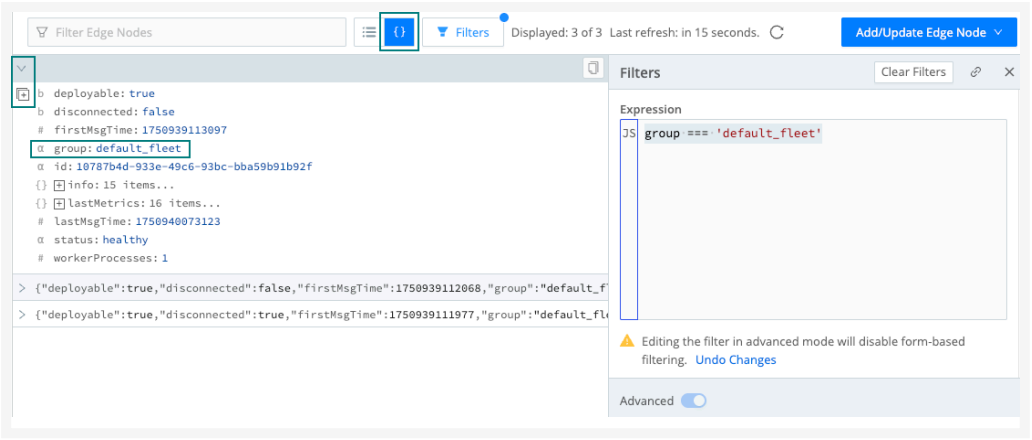
```
parent.child.grandchild === 'value'
```

For example:

```
info.aws.type === 't2.medium'.
```

You can also automatically populate the Advanced filter **Expression** field:

1. Select the **JSON view** button.
2. Expand the Node row and select the Node property, then any applicable child properties.
3. Type directly in the **Expression** field to fine-tune the filter.



Filter Edge Nodes in JSON view

Supported Edge Node Properties

You can filter on these Edge Node properties using a JavaScript expression. Remember to separate the parent and child properties using a period `.`.

```
"deployable": "boolean (Whether the Node can be deployed)",
"disconnected": "boolean (Whether the Node is connected to the Leader)",
"workerProcesses": "number (Number of Worker Processes for this Node"),
"id": "string",
"status": "string (Healthy)",
"group": "string (Fleet name)",
"info": {
  "hostname": "string",
  "platform": "string (OS Platform; linux, win32 etc.)",
  "architecture": "string (CPU Architecture; arm64, x64 etc.)",
  "release": "string (OS Kernel release)",
  "cpus": "number (Number of CPU cores)",
  "totalmem": "number (Memory in bytes)",
  "node": "string (NodeJS version)",
  "cribl": {
    "startTime": "number (Unix time in msecs when the process started)",
    "guid": "string (unique instance ID)",
    "installType": "string",
    "tags": "string array (tags configured on the Node)",
    "version": "string (Edge version)"
  },
  "freeDiskSpace": "number (Free disk space in bytes)",
  "totalDiskSpace": "number (Total disk space in bytes)",
  "kube": {
    "source": "string (always cluster)",
    "node": "string (Node name where this Pod is running)",
    "pod": "string (Pod Name)",
    "namespace": "string (namespace, always cribl)",
    "owner": {
      "kind": "string (DaemonSet)",
      "name": "string (cribl-edge)"
    },
  "conn_ip": "string (this is the IP that Cribl Leader sees; different from actual
  "localTime": "number (local Unix time on the Node in msec)"
  },
},
"lastMsgTime": "number (time when Node sent the last message in msec)",
"firstMsgTime": "number (time when Node sent the first message in msec)",
"nodeUpgradeStatus": {
  "timestamp": "number (time when Node entered the state below in msec)",
  "state": "number",
  "skipped": "number (if set, this Node is skipped by thr Leader's automatic upgrac
`
```

```
}
"lastMetrics": {
  "edge.dropped events": "number (events dropped by Node)",
  "health.inputs": "number",
  "health.outputs": "number",
  "total.in_events": "number (total events ingested by Node)",
  "total.out_events": "number (total events sent out by Node)",
  "total.dropped_events": "number (total events dropped by Node)",
  "total.in_bytes": "number (total ingested bytes)",
  "total.out_bytes": "number (total bytes egressed)"
}
```

Copy

Example Advanced Filter Expressions

You can use these example expressions as a starting point to build your advanced filters.

Edge Nodes with >10 CPUs and >8 GB RAM

Because total memory is reported in bytes, we need to calculate 8GB to bytes before using it in the filter.

```
info.cpus >= 4 && info.totalmem >= 8388608000
```

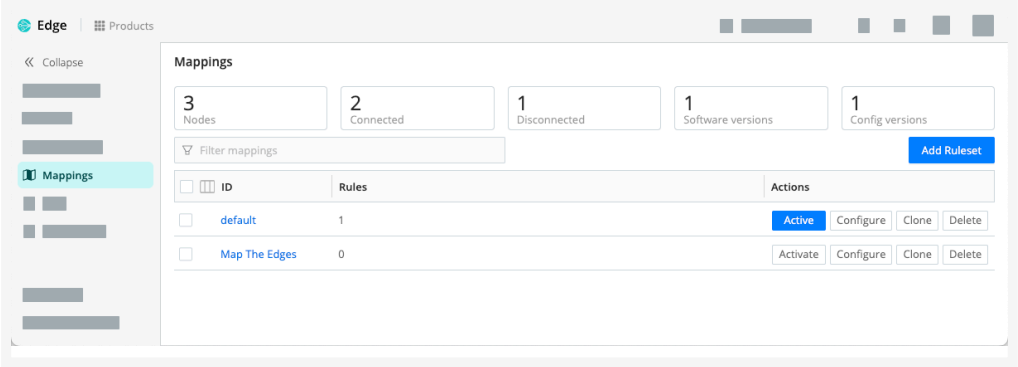
8.2.4. Map Edge Nodes To Fleets

Learn about Fleet Mappings and how to set up a Mapping Ruleset.

Fleet Mappings enable you to assign Edge Nodes to Fleets by applying specific rules defined in Mapping Rulesets. You can map Edge Nodes into Fleets based on geographical location, environment type, operating system, host, and more. This makes centralized configuration management easier for groups of Edge Nodes, and makes it easy for you to add and remove Edge Nodes from Fleets as your deployment scales.

Access Fleet Mappings

Select **Mappings** in the sidebar to display status and controls for the active Mapping Ruleset.



Mappings status/controls

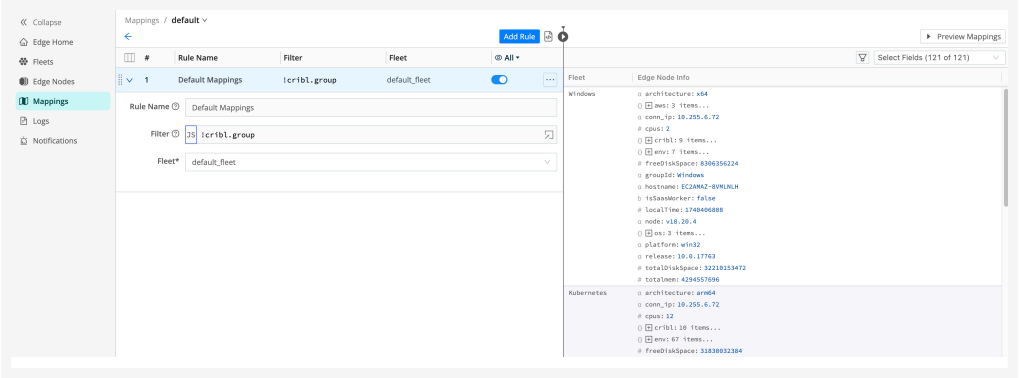
This page displays a maximum of 10,000 results.

Mapping Rulesets

A Distributed deployment of Cribl Edge starts off with a predefined `default` Mapping Ruleset.

Within a Ruleset, a list of rules evaluate Filter expressions on the information that Edge Nodes send to the Leader.

The order of rules in a Ruleset matters. The **Filter** section supports full JS expressions. The Ruleset matching strategy is first-match, and one Edge Node can belong to only one Fleet.



Managing Ruleset page

Only one Mapping Ruleset can be active at any one time, although a ruleset can contain multiple rules. The system should contain at least one defined and present Fleet.

Create a Mapping Ruleset

To create a Mapping Ruleset:

1. In the sidebar, select **Mappings**.
2. Select **Add Ruleset**.
3. Give the **New Ruleset** a unique **ID** and select **Save**.

   The **ID** for a new Ruleset can contain only letters, numbers, underscores, and dashes. Older Ruleset IDs might include other characters.

4. Select **Configure** and start adding rules with **Add Rule**.

As you build and define rules, the Preview pane will display how Edge Nodes will map to Fleets with your defined rules.

You must activate a Ruleset before the Leader can use it. To activate it, go to **Mappings** and select **Activate** on the required Ruleset. You can also **Configure** or **Delete** a ruleset, or **Clone** a Ruleset if you'd like to work on it offline, test different filters, and so on.

Although not required, you can configure Edge Nodes to send a preferred Fleet name with their payload (found in payload's `cribl.group` key). See Mapping Order of Priority for how this ranks in mapping priority.

Mapping Rule Examples

See some common Mapping Rule examples.

Map Edge Nodes Based on OS

Fleets are not aware of the operating systems their Edge Nodes run on, so they can't take into account limitations such as platform-specific Sources and Destinations.

When you manage Edge Nodes on different systems, Cribl recommends creating OS-specific Fleets.

To map Edge Nodes to Fleets based on their OS, use the `platform` filter.

OS	Filter
Linux	`platform === "linux"`
Windows	`platform === "win32"`
macOS	`platform === "darwin"`

Map Edge Nodes Connected to an Outpost

To map Edge Nodes that connect to an Outpost in a specific Outpost Group, filter on the `outpost.groupname` field, for example:

Filter: `outpost.groupname === 'EU_2'`.

You can also map Edge Nodes by the specific Outpost they connect to by filtering on the `outpost.guid` or `outpost.host` fields, for example:

Filter: `outpost.host === '5ab6c676be6a'`.

Map Edge Nodes Based on Tags

You can use tags assigned to Edge Nodes to map specific Nodes to a Fleet.

To map Edge Nodes that have a specific tag to a Fleet, filter on the `cribl.tags` field, for example:

Filter: `cribl.tags.includes("WinLaptop")`

Find Edge Nodes Running in a Container

Edge Nodes running in a container report additional metadata about the container to the Leader. This helps you quickly identify which of your Edge Nodes are running in a container from the Leader UI and map them to the appropriate Fleets.

When Cribl Edge is running in a container, you will see the `hostOs` property in events collected by the System State Source. This object will contain properties about the container when they're available, such as:

- `arch`
- `cpu_count`
- `cpu_speed_mhz`
- `cpu_type`
- `interfaces`
- `memory`
- `platform`
- `release`
- `hostname`
- `timezone`

From the **Mappings** menu, you will also see the `hostOs` property in the **Edge Node info** panel. This object will include the `id`, `addresses`, and `version` of the container when available.



`interfaces` is a set of key-value pairs that contain an array of CIDR values.

Complex Edge Node Mappings

The mapping filter supports JS expressions and lets you string together more complex filters.

For example, let's assume that you want to define a rule for all hosts that satisfy this set of conditions:

- IP address starts with `10.10.42`, AND Edge Node has more than 6 CPUs, OR:
- `CRIBL_HOME` environment variable contains `DMZ`.

Filter: `(conn_ip.startsWith('10.10.42.') && cpus > 6) || env.CRIBL_HOME.match('DMZ').`

Collect AWS EC2 Instance Tag Metadata

If Cribl Edge is connected to an AWS EC2 instance, you will see metadata for AWS instance tags in the **Mappings** tab. You can use tags to map EC2 Nodes to the correct Fleet.



For Cribl Edge to obtain tags, you **must** allow access to tags in the EC2 instance metadata.

Mapping Order of Priority

Priority for mapping to a Fleet is as follows: Mapping Rules > Fleet (`cribl.group`) sent by Edge Node > `default_fleet` Fleet.

- If a Filter matches, use that Fleet.
- Else, if an Edge Node has a Fleet defined, use that.

9. Workspaces

Workspaces offer a multi-tenancy capability, enabling you to create multiple isolated instances in your Cribl.Cloud Organization. This lets you strengthen your security, and fulfill compliance and isolation requirements.

Each Workspace offers a dedicated virtual private cloud (VPC) that acts as a separate environment within your Organization. Workspaces do not share product configurations, resources, or data flows, and each has a separate Leader Node. They do share a layer of access control (Organization-level Permissions and SSO configuration), as well as API credentials, licenses, and billing.

A centralized interface allows you to manage all your Workspaces and control access for Members and Teams to individual Workspaces.

Multiple Workspaces require a Cribl.Cloud Enterprise plan. For details, see Pricing.

An example use case scenario for creating multiple Workspaces is setting up separate environments for different business units in an enterprise. Data management, security, development, and any other units can have their own federated Workspaces, with completely separate members and permissions lists. This way, they can work in isolation without risk of interference and with increased focus.

Limitations

You can create up to 5 Workspaces per Organization.

If you need more Workspaces in your Organization (up to 50), contact Cribl Support to discuss extension options.

Multiple Workspaces require a Cribl.Cloud Enterprise plan. For details, see Pricing.

Manage Workspaces

To list the Workspaces in the Organization, in the sidebar, select **Workspace** or **Organization** and then **Manage Workspaces**. The list includes the name, ID, description, Cribl Insights retention status, tags, Cribl version number, and current state for each Workspace in the Organization.

Select a Workspace name or the ellipsis (**. . .**) in the **Actions** column to edit the Workspace name, description, and tags or delete the Workspace.

View Workspace Details

Workspaces provide a set of information that you can use when configuring data flow through their Edge Nodes.

View Access Details

To view detailed access information of your Workspace, in the sidebar, select **Workspace** and then **Access**.

This page presents a summary of information about your Cribl.Cloud Organization, as well as the **Cribl.Cloud URL** for the current Workspace.

Cribl.Cloud URL

Cribl.Cloud URL is a static address associated with the load balancer that is in front of the Leader. Customer-managed (hybrid) Edge Nodes will connect to this address on port 4200, while the Leader UI is served from this address on port 443.

You can use this URL for certain API calls and certain Collection operations coordinated by the Leader.

Static External IPs for the Leader

The **Leader NLB IPs** field in **Workspace** > **Access** lists the IPs for the Leader Network Load Balancers associated with the Workspace.

Typically, only 2 of the 3 addresses will be active (returned by DNS) at any time, while the inactive address is swapped in during infrastructure maintenance events.

You can add all those IPs to your firewall allowlist to ensure that the load balancers are accessible across your hybrid deployment.

Stream Worker Group Details

Stream Worker Group Details provides information about Cribl-managed Worker Groups in Cribl.Cloud, if any are provisioned.

If no Worker Group is provisioned, you can select **Provision Now** to provision infrastructure for the Group. After a lag, the Group will be ready to process data, and this modal's remaining fields will populate.

For a provisioned Worker Group, you see the following information:

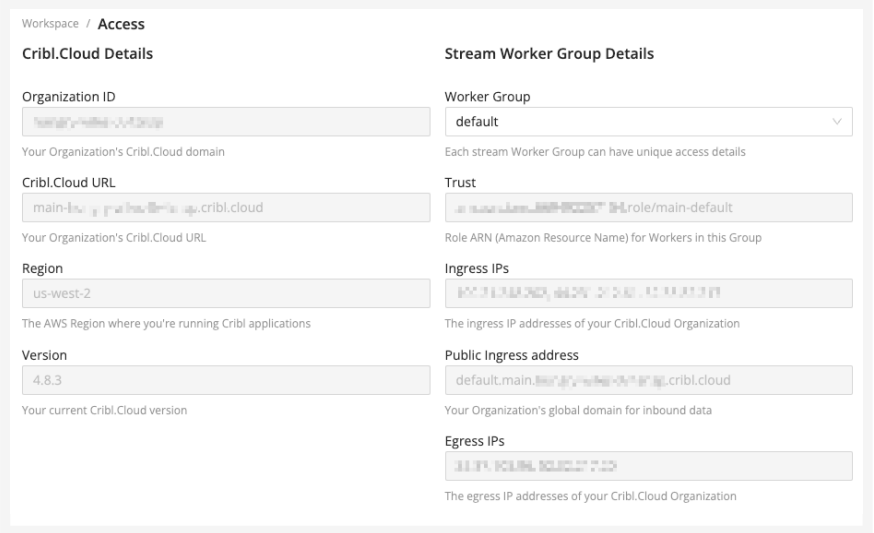
Field	Description
Trust	AWS IAM role ARN for Cribl-managed Workers in this Worker Group. Use this ARN as the principal in AWS trust policies when configuring cross-account access.
Ingress IPs	The IPv4 addresses of Worker Groups' load balancers, also used when receiving data from Push Sources. These addresses will remain constant, so you can build firewall rules around them. Three Ingress IPs are provided for each Worker Group, one on each Availability Zone, assuming the Group has at least three Workers. In a Worker Group of fewer than three Workers, one of the IPs will be inactive.
Public Ingress address	Each Group's domain for inbound data. This address prepends the Group name to the Organization's global domain name. It does not append ports per data type - you can obtain these from the Data Sources tab.
Egress IPs	Your Cribl.Cloud Organization's current public IP addresses. These addresses are Group-specific and also dynamic: Cribl will occasionally update them when we need to rescale core infrastructure.

Public Ingress address and **Egress IPs** are used for both outbound connections from the Workers and Pull Sources.

Because individual **Ingress IPs** can become inactive, Cribl recommends that you send your data to the **Public Ingress address** instead of directly to IPs.

Egress IPs are not static.

If you need a static egress IP, contact Cribl Support.



Access Details modal for a provisioned Group

Configuring Stream Groups (beyond the `default` Group) requires a certain plan. For details, see Pricing. For details about creating and provisioning Groups, see Cribl.Cloud Worker Groups.

View Data Sources

To view detailed access information about a Workspace's Data Sources, in the sidebar, select **Workspace** and then **Data Sources**.

The **Data Sources** page lists ports, protocols, and data ingestion inputs that are open and available to use, including pre-enabled Sources. Use the **Group** drop-down to filter these details per Cribl-managed Worker Group in Cribl.Cloud. For details, see Available Ports and TLS Configurations.

For each existing Source listed here, Cribl recommends using the preconfigured endpoint and port to send data into Cribl Stream.

Get ARN

To get the Worker role ARN for a Workspace, in the sidebar, select **Workspace**. In the **Stream Worker Group Details** section, select a Worker Group and copy the value of the **Trust** field.

You can use this **Worker ARN** as the principal in the trust policy for an AWS account's IAM role that Cribl Workers will assume. Attaching this trust relationship enables the `AssumeRole` action, providing cross-account access. For usage details, see the AWS Cross-Account Data Collection topic's **Account B Configuration** section.

This option applies only to your Cloud Workers. You cannot use this technique to enable access to customer-managed hybrid Workers.

Set Up ACL

To set up Access Control List (ACL) Rules, in the sidebar, select **Workspace** and then **Access Control List**.

ACL Rules (IPv4 CIDR ranges) let you restrict data sent to your data sources. The Rules you define here are global to all Cribl-managed Worker Groups in Cribl.Cloud in the current Workspace. You can set up a maximum of 6 ACL Rules.

The default `0.0.0.0/0` rule (modifiable) imposes no limits. End a rule with `/32` to specify a single IP address, or with `/24` to enable a whole CIDR block from `x.x.x.0` to `x.x.x.255`.

Select **Save** after adding, modifying, or removing rules. Each change takes up to 5 minutes to propagate. During this time, rule editing is temporarily disabled to prevent conflicts. A successful update proceeds silently - you will not see a confirmation message.

- The **ACL** options apply only to Cribl-managed Workers. You cannot use them to set access rules on customer-managed hybrid Workers.
- If you delete the default `0.0.0.0/0` rule and do not configure any other ACL rules, all inbound data to Cribl-managed Worker Groups in this Workspace will be blocked until you add an allow rule.

9.1. Configure Workspaces

Learn how to configure Workspaces in your Cribl.Cloud deployment.

Before You Begin

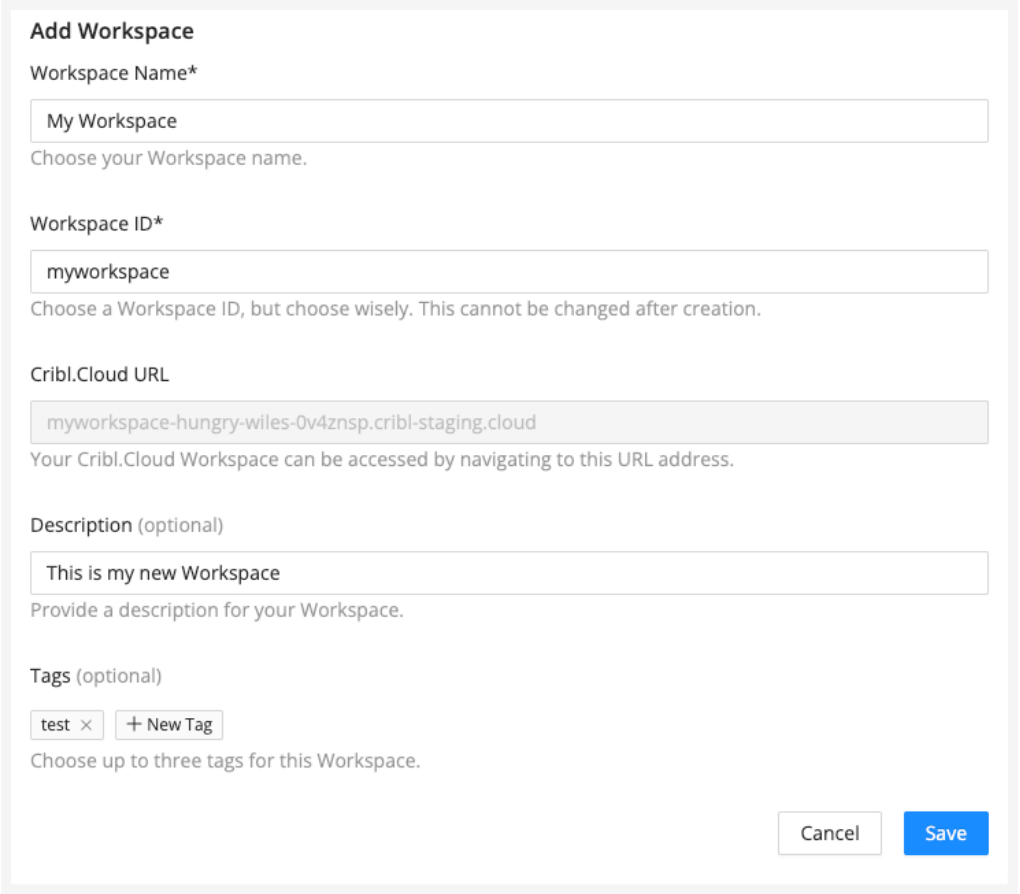
When you create a new Workspace in Cribl.Cloud, Cribl automatically provisions Cribl-managed Worker Nodes in the `default` Worker Group. **These Worker Nodes consume infrastructure credits**, but you can deprovision the `default` Worker Group from Worker Group settings. See Manage Cribl.Cloud Worker Groups for steps.

Add New Workspace

To create a new Workspace:

1. On the top bar, select your current Workspace name. This opens a modal with a list of all Workspaces you have access to.
2. Select **Add Workspace**.
3. Fill the form with the following information:

Field	Description	Notes
Workspace Name	Human-readable name for the Workspace.	
Workspace ID	Internal ID for the Workspace. This ID will be part of the Workspace's Cribl.Cloud URL.	**Cannot be changed** once the Workspace is created.
Description	Additional Workspace description.	Optional
Tags	Up to 3 tags you can use to filter Workspaces.	Optional



Creating a new Workspace

4. Confirm with **Save**.

The Workspace can take up to several minutes to spin up. While the process continues, you can navigate away and keep working with your Cribl.Cloud instance as usual.

Edit or Delete a Workspace

To edit a Workspace:

1. On the top bar, select your current Workspace name. This opens a modal with a list of all Workspaces you have access to.
2. Next to the Workspace you want to edit, select **Settings** in the **Actions** column to open the **Details** page for the Workspace.
3. On the **Details** page, you can change the name of the Workspace, description, and tags. However, you can't change the Workspace ID after it has been created.
4. Select **Save**.

For all Workspaces except `main`, you can select **Delete Workspace** on the **Details** page to delete it. You will need to manually type the name of the Workspace as a security measure.

10. Administration

Consult the following topics to learn how to manage and configure your Cribl Edge deployment.

- FinOps Center
- Upgrade Cribl Edge
- Notifications
- Licensing
- Uninstall Cribl Edge from Linux
- Uninstall Cribl Edge from Windows
- Uninstall Cribl Edge from macOS
- Customize the Interface
- Scripts

10.1. Upgrade Cribl Edge

Learn about the ways you can upgrade Cribl Edge, as well as general and version-specific upgrade considerations.

When upgrading a Cribl Edge deployment, the general order of steps is:

1. Upgrade the Leader Node.
2. Upgrade Outpost Nodes, if present.
3. Upgrade Edge Nodes.
4. Commit and deploy the changes on the Leader.



If your deployment uses a second Leader, refer to Upgrading with a Second Leader.

Prepare for an Upgrade

Before upgrading the Leader, plan your whole upgrade strategy, in particular whether you want to upgrade all your Edge Nodes at the same time. Edge Nodes will not upgrade automatically when you upgrade the Leader. In particular:

- Revise Better Practices for things to consider before starting the upgrade.
- Ensure you have configured Backup and Rollback.
- If you are upgrading older Cribl Edge versions, pay attention to the special cases described in Troubleshooting Edge Upgrades.

Supported Version Differences Between Leader and Edge Nodes

Cribl strongly recommends keeping Leader Nodes and Edge Nodes on the same version for the best experience and to ensure access to all features.

If you need to keep Edge Nodes on a different version than the Leader, for example during a phased upgrade, make sure they do not stay on a significantly older version.

Do not upgrade Edge Nodes to a newer version than the Leader Node.

Upgrade from Non-GA Versions

Cribl Edge does **not** support direct upgrades from any pre-GA version (such as a Cribl-provided test candidate) to a GA version. To get the GA version running, you must perform a new install.

10.1.1. Upgrade Leader Node

Upgrade the Leader Node of your Distributed deployment of Cribl Edge.

The Leader Node, which controls both Edge Nodes and Stream Workers, upgrades differently depending on your environment:

- In Cribl.Cloud, your Leader Node will be upgraded automatically as Cribl releases new product versions.
- In an on-prem deployment, you select when you want to upgrade your Leader Node.

There are two ways to upgrade the Leader Node in an on-prem deployment: directly from the UI, and manually from the command line.

Before you start upgrading your Leader Node, make sure to read Prepare for an Upgrade.

If your deployment uses a second Leader configured for failover, refer to Upgrade with a Standby Leader.

Upgrade Leader from the UI

To upgrade an on-prem Leader Node directly from the UI, go to **Settings** > **Global** > **System** > **Upgrade**.

You can choose to either use upgrade packages from the Cribl content delivery network (CDN), or point to packages located in a specified remote or filesystem location:

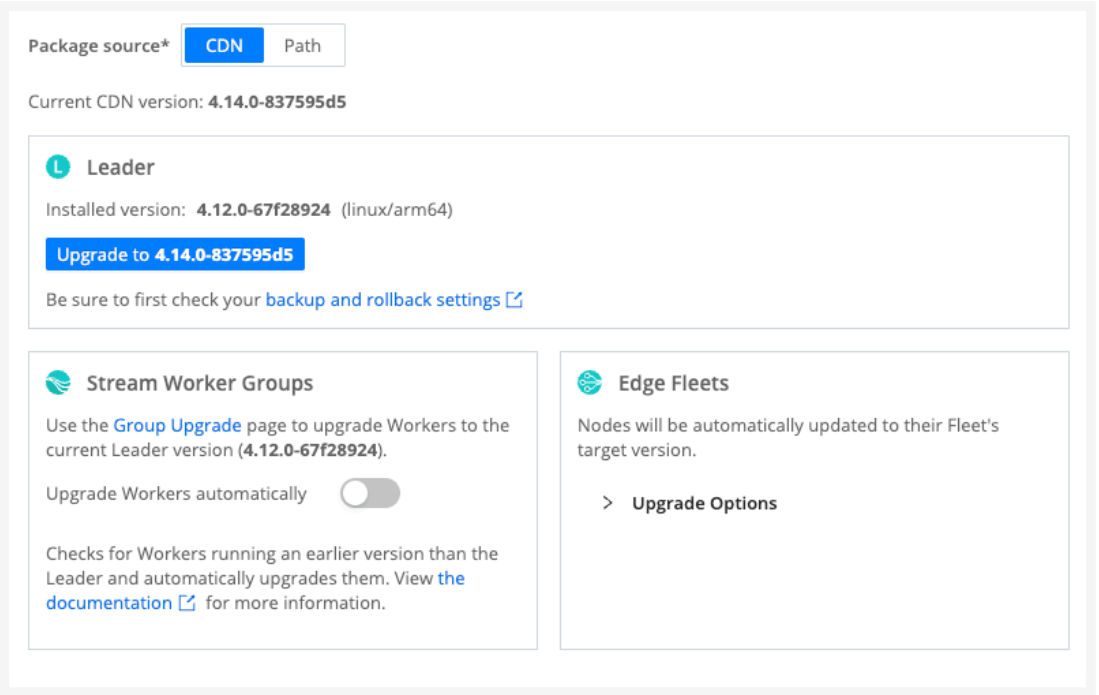
Package Source	Description
CDN	Downloads an installation package directly from the Cribl content delivery network. The Leader will always upgrade to the current CDN version when you select **Upgrade**.
Path	Uses an installation package from the filesystem or a remote URL. You must provide the correct path for your Leader platform and architecture.

If your Leader controls Stream Workers, upgrading it can cause Workers to upgrade automatically, depending on configuration. Refer to Automatic Upgrades for more information.

Upgrade Leader from CDN

To upgrade a Leader Node using packages hosted in Cribl CDN:

1. Go to **Settings** > **Global** > **System** > **Upgrade**.
2. Select **CDN** in **Package source**.



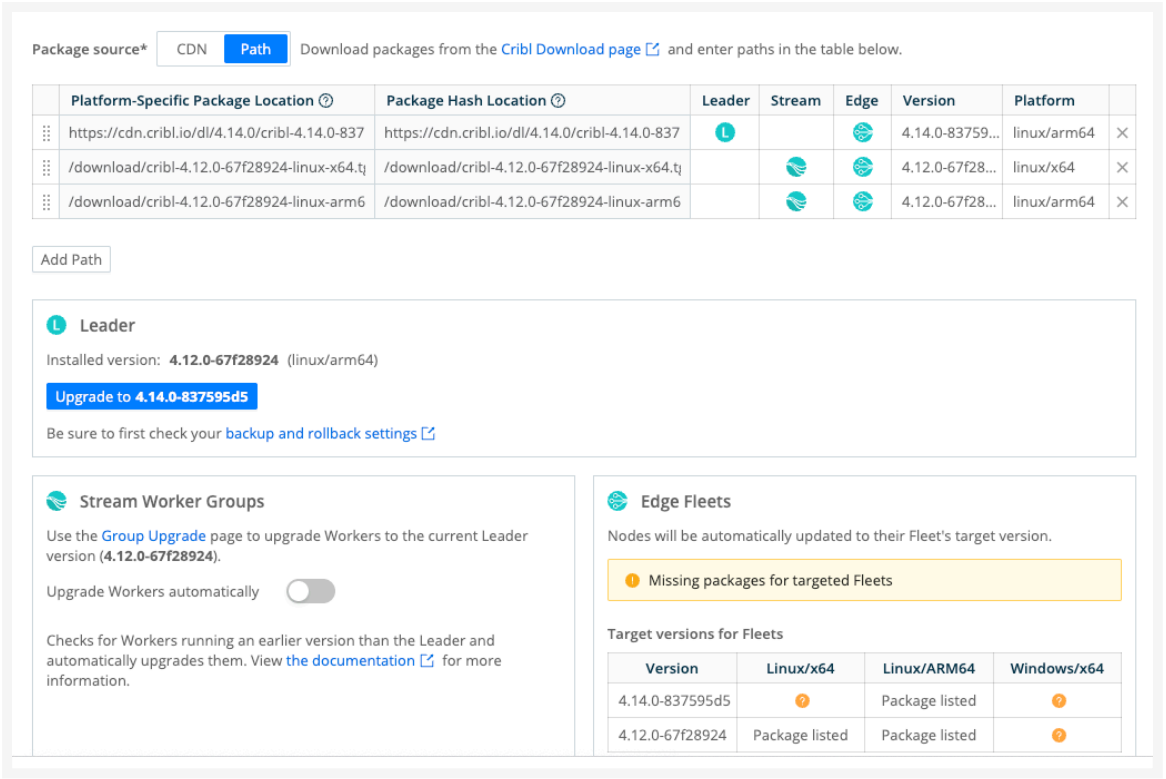
Upgrade the Leader using CDN

   If your Leader is running an older version, you will see the latest version of the Cribl package that is available on the CDN.
3. Upgrade the Leader by selecting **Upgrade to <version>**.

Upgrade Leader from a Path

To upgrade a Leader Node using specified packages:

1. Go to **Settings** > **Global** > **System** > **Upgrade**.
2. Select **Path** in **Package source**.



Upgrade the Leader using Path

3. Provide the Location of a Cribl package in **Platform-Specific Package Location**. You can either:

   - Point to an HTTP URL that houses the package, for example `https://cdn.cribl.io/dl/4.14.0/cribl-4.14.0-837595d5-linux-x64.tgz`, or:
   - Download the package from the Cribl download page and point to local filesystem path, for example `/download/cribl-4.14.0-837595d5-linux-x64.tgz`. In the second case, the Leader must have access to the local filesystem location. When you stage these packages on your own servers, preserve the original file names.

Make sure you download the correct package for your platform and architecture.

4. Download the package's SHA or MD5 hash, or point to its URL in **Package Hash Location**. For a CDN location, you can append `.sha256` to the contents of the **Platform-Specific Package Location** field. The **Leader** column in the table will confirm with an icon when the package will be used to upgrade the Leader.

When Cribl Edge is running in FIPS mode, MD5 is not an acceptable hashing algorithm.

5. When you have added the correct package for the Leader, select **Save**.
6. Select **Upgrade to <version>** in the **Leader** section.

Additional Packages for Node Upgrades

You can use the **Platform-Specific Package Location** table to specify packages for different versions and platforms/architectures for use when upgrading Stream Workers and Edge Nodes.

The **Stream** and **Edge** columns in the table will indicate which packages will be used to upgrade which Nodes in your deployment.

Missing Package Links

When you see a warning about "Missing packages", it means there are Worker Groups or Edge Fleets that can't be upgraded due to a missing package path.

To add package links:

1. Select the **Copy** icon to add the link to your clipboard.
2. Paste the copied package link into a new **Platform-specific package location** field.
3. Add the appropriate path directory to the beginning of the installation package path (where your Cribl upgrade packages are located).
   For example: `/download/cribl-4.14.0-837595d5-linux-x64.tgz`
4. Select **Save** to add the upgrade package path and resolve the warning.

Upgrade Leader Manually

Besides upgrading from the UI, you can also upgrade your Leader Node manually, using the CLI. This might be desired in, for example, airgapped deployments, or when you want to select a Leader version lower than the latest available.

To upgrade the Leader Node manually:

1. Download the Cribl package in your target version, platform, and architecture from the Cribl download page.
2. Commit and deploy your current Leader configuration.
   Optionally, `git push` to your configured remote repo.
3. Stop the Leader.
   - Optional but recommended: Back up the entire `$CRIBL_HOME` directory.
   - Optional: Check that the Workers and Edge Nodes are still functioning as expected. In the absence of the Leader Node, they should continue to work with their last deployed configurations.
4. Uncompress the new Cribl package on top of the old one.
5. Change ownership of the `cribl` directory as needed:

```
sudo chown -R cribl:cribl $CRIBL_HOME
```

Copy

6. Restart the Leader and log back in.
7. Wait for all the Workers and Edge Nodes to report to the Leader, and ensure that they are correctly reporting the last committed configuration version.

Leader Upgrades and Edge Nodes

Edge Nodes will not upgrade automatically when you upgrade the Leader, even if set to current Leader version. See Upgrade Edge Nodes in a Fleet to learn how to upgrade all Edge Nodes in a Fleet.

Upgrade with a Standby Leader

For deployments with a standby Leader configured for failover, follow this upgrade order:

1. Stop the standby Leader first.
2. Upgrade the standby Leader.
3. Start the standby Leader.
4. Stop the primary Leader; the upgraded standby will take over automatically through the HA failover mechanism.

10.1.2. Upgrade Outpost

Use Outpost Group configuration to automatically upgrade Outposts.

Required Permission: **Admin/Owner**

Outposts are upgraded in bulk by setting the desired target version for the Outpost Group they belong to.

To automatically upgrade Outposts in a group:

1. In the sidebar, select **Outposts**.
2. In the row for the Outpost Group you want to upgrade, select the button in the **Target Software Version** column.
3. In the **Target Version** modal, select a version from the **Upgrade target version** dropdown. You have two options:
    - **None** - when set to None, the Outposts in the group will never upgrade automatically.
    - **<version> (current leader version)** - when set to this option, the Outposts will be upgraded to match the software version of the currently running Leader Node.
4. Confirm with **Save**.

The next time the Outpost in the selected group connects to the Leader, it will upgrade to the selected version.

Outposts in the selected group will not upgrade to any future version until you change the target software version in the drop-down (even if you upgrade the Leader).

The **Add/Update Outpost Node** command uses the **Target Software Version** for your Outpost Group when constructing the script for upgrading your Outposts.

See Add an Outpost with a Bootstrap Script for more details about the command.

10.1.3. Edge Node Upgrades

Upgrade Edge Nodes automatically or manually.

There are two ways to upgrade Edge Nodes: in bulk together with the whole Fleet, and manually, per Edge Node.



Edge Nodes in containers undergo a different upgrade process. Refer to Upgrading Containers for more information.

Fleet Upgrades

Upgrading Edge Nodes in bulk with their Fleet is the more straightforward of the two upgrade options, and manageable completely from the Leader Node UI. With this method, you set the desired target version for all Edge Nodes in a given Fleet. All Edge Nodes will then upgrade automatically to the target version.

See Upgrade Edge Nodes in a Fleet for more details.



Upgrading Edge Nodes from the Leader is available on certain plan/license tiers. For details, see Pricing.

Automatic Edge Node upgrades have the following limitations:

- You can't automatically upgrade Edge Nodes installed via RPM or from a .deb package. Refer to Upgrade RPM Installations and Upgrade .deb Installations for the manual upgrade process for those Nodes.
- You can't automatically upgrade Edge Nodes on Windows running version 4.3.0 or older. Use manual upgrade instead.
- Upgrading a Single-instance deployment requires a manual upgrade.

Manual Upgrades

Upgrading Edge Nodes manually requires running the upgrade process on the CLI for each Node.

Manual upgrades let you pause and resume upgrades as needed, which gives you more granular control over the process. You can also upgrade individual Nodes separately. This is useful when you're testing upgrades on a subset of Nodes before rolling out to the entire deployment.

Consider upgrading via CLI when your deployment is airgapped for security or compliance reasons and can't download new packaged from the internet.

See Upgrade Edge Nodes Manually for more details.



When upgrading Edge Nodes on Windows, consult Debugging Cribl Edge on Windows.

10.1.3.1. Upgrade Edge Nodes In A Fleet

Upgrade a whole Fleet of Edge Nodes by defining a target version.

To automatically upgrade Edge Nodes in one Fleet, you can set the desired target version for the Fleet through the UI.



Upgrading Edge Nodes from the Leader is available on certain plan/license tiers. For details, see Pricing.

Automatic Fleet upgrades have the following limitations:

- You can't automatically upgrade Edge Nodes installed via RPM. Refer to Upgrade RPM Installations for the manual upgrade process for those Nodes.
- You can't automatically upgrade Edge Nodes on Windows running version 4.3.0 or older. Use manual upgrade instead.
- Automatic upgrades on Edge Nodes on Windows require specific configuration of service accounts. See Service Accounts and Edge Node Upgrades for more information.
- Upgrading a Single-instance deployment requires a manual upgrade.

Upgrade Edge Nodes

To automatically upgrade Edge Nodes in a Fleet:

1. In the sidebar, select **Fleets**.
2. In the row for the Fleet you want to upgrade, select the button in the **Target Software Version** column.
3. In the **Target Version** modal, select a version from the **Upgrade target version** dropdown. You have two options:
    - **None** - when set to None, the Nodes in the Fleet will never upgrade automatically. Choose this option for highly critical Edge Nodes or Nodes managed by other departments that require strict control over upgrades.
    - **<version> (current leader version)** - when set to this option, the Nodes will be upgraded to match the software version of the currently running Leader Node.
4. Confirm with **Save**.

The next time the Node in the selected Fleet connects to the Leader, it will upgrade to the selected version.

Nodes in the selected Fleet will not upgrade to any future version until you change the target software version in the drop-down (even if you upgrade the Leader).

The **Add/Update Edge Node** command uses the **Target Software Version** for your Fleet when constructing the script for upgrading your Edge Nodes. For details, see Edge Node Updates Using Fleet Version Settings.

Subfleet Target Version

Subfleets do not inherit the **Target software version** from their parent Fleet, but you can set the target software version for each Subfleet using the same steps you used in the parent Fleet.

Package Sources for Offline Upgrades

Automatic Edge Node upgrade in Fleets will download packages from the Cribl CDN. For offline use in an on-prem deployment, you can also pre-download packages for specific platforms and architectures. Make sure you place the packages in a location that the Leader Node can access. To indicate that the system should use those packages:

1. Go to **Settings** > **Global** > **System** > **Upgrade**.
2. Select **Path** in **Package source**.
3. In **Platform-Specific Package Location**, enter the path to your locally downloaded package. You can add multiple paths to target different platforms or target Cribl Edge versions.
4. In **Package Hash Location**, point to the SHA or MD5 hash file for each of the packages.
5. Select **Save**.

When the Fleet next upgrades, it will use those downloaded packages instead of CDN.

Enable Legacy Edge Upgrades (Deprecated)

While this functionality still exists, it is deprecated and we do not recommend using it.

Enabling legacy upgrades removes the ability to automatically upgrade Nodes on a per-Fleet basis.

If you experience issues with the Fleet upgrade functionality, this option lets you use the job-based upgrade framework and manually upgrade Fleets via the Fleet Upgrade UI and **Upgrade** button.

To enable legacy upgrades, go to **Settings** > **Global** > **System** > **Upgrade**, and in **Edge Fleets**, under **Upgrade Options**, select **Enable Legacy Edge upgrades**. You'll also need to toggle **Disable Jobs/Tasks** off. See Jobs and Tasks.

10.1.3.2. Upgrade Edge Nodes Manually

Upgrade individual Edge Nodes using CLI.

Upgrading individual Edge Nodes manually via the CLI gives you more control over the process than automatic Fleet upgrade, letting you pause and resume upgrades as needed and select specific versions to upgrade to.

To upgrade an individual Edge Node manually:

Prepare the Installation

1. Download the Cribl package from the Cribl download page. Pay attention to select the correct package for your platform and processor type. If the Leader is on a prior release, see Cribl Past Releases section at the bottom of the page for older packages.
2. Stop the Edge Node.

Install the New Edge Node

Linux Windows macOS

1. Uncompress the new Edge version on top of the old one, for example in the `/opt/cribl-edge` directory:

```
tar xvzf cribl-<version>-<build>-<arch>.tgz
```

Copy

Finalize Installation

To finalize the installation:

1. Restart Cribl Edge.
2. Ensure that newly upgraded Edge Nodes report to the Leader with their new software version.
3. Commit and deploy the newly updated configuration **only after all** Edge Nodes have upgraded.

Your Edge Node upgrade is now complete.

Upgrade RPM Installations

Manual upgrade is the only possible way of upgrading Edge Nodes install using RPM packages.

To upgrade an RPM installation:

1. Download the new Cribl Edge RPM package for your desired version.
2. Use your system's package manager (such as `yum` or `apt-get`) to upgrade the existing Edge Node:

```
sudo yum update <path-to-RPM-package>
```

Copy

3. If you are upgrading the Edge Node from an older version to 4.16.0 or later, set optional Linux capabilities in the Cribl Edge service file located at `/etc/systemd/system/cribl-edge.service` or `/lib/systemd/system/cribl-edge.service`:

```
[Service]
AmbientCapabilities=CAP_DAC_READ_SEARCH CAP_SYS_PTRACE
```

Copy

The upgrade retains all changes you make to environment variables.

For large organizations, Cribl recommends establishing an internal repository for RPM packages. This centralized approach streamlines upgrades by eliminating the need to download individual RPMs to each node, improving efficiency and consistency.

Upgrade .deb Installations

Manual upgrade is the only possible way of upgrading Edge Nodes installed using .deb packages.

To upgrade a .deb installation:

1. Download the new Cribl Edge .deb package for your desired version.
2. Use the apt package manager to upgrade the existing Edge Node:

```
sudo apt install <path-to-new-deb-package>
```

Copy

The upgrade retains all changes you make to environment variables.

10.1.4. Upgrade Containers

Learn how to upgrade Edge Nodes on Containers.

Upgrading Cribl Edge Containers

If you're asking, "Why didn't my Kubernetes Node upgrade?" or perhaps, "Why *did* my Docker Node upgrade?", then you're in the right place.

Cribl Edge handles upgrades for containerized Nodes (4.5.0 and newer) differently than standalone or on-prem Nodes. Leader Nodes don't upgrade Edge Nodes that are running in a container (such as containerd, Docker, or Kubernetes), because Edge Nodes revert back to their Helm chart image after a Pod restarts. Instead, upgrade containerized Nodes manually.

Upgrades for containerized Nodes also work a little differently in 4.5.0 and newer compared to 4.4 and older. Consider these upgrade scenarios:

Scenario	Do the Nodes Upgrade?	Logic Explanation
A Fleet contains **Kubernetes / containerized** Nodes running version v.4.5.0. The Fleet has a set **Target Software Version** of 4.5.2.	No	• The Nodes are running in a container. Containerized (Docker, Kubernetes, etc) Nodes will not be upgraded automatically. • Nodes on version 4.5.0 and newer inform the Leader that they are containerized Nodes.
A Fleet contains **Kubernetes / containerized** Nodes running v.4.4. The Fleet has a set **Target Software Version** of 4.5.0.	Yes	• Nodes on v.4.4 and older do not have the ability to inform the Leader that they're containerized.

Specific Considerations for Upgrading Containers

When upgrading, you must consider the initial install method for Edge Nodes relevant to containerized deployments.

Considerations for Kubernetes (Helm Charts)

Recommended method: If you initially deployed these Edge Nodes using Helm Charts, it's recommended to maintain consistency and upgrade them using the same method.

Process: Update the Helm Chart version to the desired Cribl Edge version in your Helm repository.

Deploy the updated Helm Chart to your Kubernetes cluster. Kubernetes will manage the upgrade process and ensure no downtime due to restarts.



Upgrading Edge Nodes manually within containers is highly discouraged. It can interfere with Kubernetes' resource management and lead to unexpected behavior.

For details on managing your Kubernetes Deployment, see Deploying via Kubernetes.

Considerations for Docker Containers

Cribl Edge upgrades won't be applied if your Node Info indicates `env.CRIBL_INSTALL_TYPE` is set to `CONTAINER`. This is because upgrades are designed for installations on physical or virtual machines, and containerized deployments manage updates differently.

Upgrading Kubernetes Edge Nodes

Edge Nodes deployed in Kubernetes can't be upgraded via the Leader. Instead, use our Helm charts to upgrade the Edge Nodes deployed in your Kubernetes cluster:

```
helm upgrade --install
```

Copy

For details on common challenges, see edge-via-kubernetes.

Updating the Docker Image

Cribl recommends that you always use our latest stable container image wherever possible. This will provide bug fixes and security patches for any vulnerabilities that Cribl has discovered when scanning the base image OS, dependencies, and our own software.

You can explicitly pull the latest stable image with this CLI command:

```
docker pull cribl/cribl:latest
```

Copy

Updating the Packaged OS

Cribl strongly recommends that you monitor and patch vulnerabilities in the packaged OS. The base OS might have been updated with fixes for new vulnerabilities discovered after Cribl published its container images. This is especially important if you choose to keep an earlier Cribl image in production.

You can update the base OS image by updating the package, as shown in the following `Dockerfile`. This example assumes you're updating Cribl's `latest` image:

```
FROM cribl/cribl:latest
RUN apt-get update && \
    apt-get -y upgrade dpkg
```

Copy



Cribl Edge supports both Docker and `containerd` runtimes.

10.1.5. Better Practices For Upgrading Cribl Edge

Streamlining the Upgrade Process.

While Cribl Edge is designed to provide an easy way to upgrade the version of Cribl Edge on your nodes, we recognize that everyone's environment and requirements are different. Here are some recommendations for planning upgrades in your environment either through our Fleet-level upgrades, or when using other deployment tools.

Plan Your Upgrade

A successful Cribl Edge upgrade requires careful planning. Consider these steps to ensure a smooth transition.

For your Environment

- Identify your current Cribl Edge version. Determine your Cribl Edge version by running the command `./cribl version` in your terminal or by checking the upper right corner of the Cribl Edge UI landing page.
- Evaluate changes or updates in your OS that could interfere with the upgrade, such as any software that might block connections. Ensure the Cribl process is allowlisted and allowed to execute without limitations to facilitate a smooth installation.
- Coordinate with your Security/IT team. Before deploying Cribl Edge on Nodes, consult with your security or IT department. They can advise on any necessary approvals or specific deployment guidelines you need to follow.

For your Deployment

- Review Supported Version Differences Between Leader and Edge Nodes.
- Review the Release Notes for the version you want to upgrade to and any previous versions you might have skipped.
- Check the Known Issues page for anything might impact your environment, including compatibility issues with your OS, third-party applications, or configurations. Additionally, assess potential issues with Sources, Destinations, and other configurations.
- If you initially installed Cribl Edge using containers, refer to the Upgrading Containers documentation for details.

Decide on an Upgrade Strategy

For organizations using Leader-initiated updates, Fleets offer a valuable tool for managing upgrades. By categorizing Edge Nodes into different Fleets based on upgrade schedules or risk tolerance, you can control the upgrade process with greater precision. If you are using other upgrade management tools, using Fleets this way can still be beneficial, but their impact might be less pronounced.

Consider organized your Fleets into:

- **Mission-critical Edge Nodes**: To prioritize software stability and minimize downtime for these critical Edge Nodes, organize them into Fleets that will follow a longer upgrade cycle.
- **Non-critical Edge Nodes**: Organize non-critical Edge Nodes into Fleets that will have a more frequent upgrade cycle (every two to three months, for example), to balance feature adoption and system stability.
- **Early-Adopter Edge Nodes**: To evaluate new releases, consider creating dedicated test Fleets comprised of Edge Nodes suitable for testing. These Fleets can be upgraded frequently.

Configure Fleets to match your desired upgrade approach. See Upgrade Edge Nodes in a Fleet for details.

Establish an Appropriate Window For Upgrades and Maintenance

Consider acceptable downtime and maintenance windows:

- Edge Node upgrades will require a restart of the Edge process. Consider organizing Fleets in a way that accounts for rolling upgrades and avoid potentially concurrent downtime.
- Note that Edge Node upgrades occur in batches of 500, so large Fleets may require longer upgrade times.
- Coordinate with Edge Node owners to schedule upgrades that minimize service interruptions, taking into account time zone variations.

Roll Out Fleet-Level Upgrades for Large Deployments

Fleet-level upgrades involve updating multiple Edge Nodes simultaneously. This approach is ideal for larger deployments where managing individual upgrades would be impractical.

Steps for Fleet-Level Upgrades

- **Staged rollout**: Start your Cribl Edge upgrade in a designated test Fleet or SubFleet (as a pilot group) to minimize risk and identify issues before impacting production. Or consider separating your dev/testing

environment from your production.

- **Asynchronous upgrades**: Use the upgrading framework to allow Nodes to request upgrade packages from the Leader in batches, reducing load on the Leader.
    - Nodes communicate to the Leader via heartbeat every 60 seconds.
    - Upgrade packages are requested in batches of 500.
- **Error handling**: Look for informative error messages in logs such as `msiexec`, the Windows Event Log, and `cribl.log`. Log files are located in the `$CRIBL_HOME/log/` directory:
    - Server/API (Leader logs): `C:\ProgramData\Cribl\log\cribl.log`: Server/API (Leader)
    - Edge Node logs: `C:\ProgramData\Cribl\log\worker\0\cribl.log`

Monitor the Upgrade Status

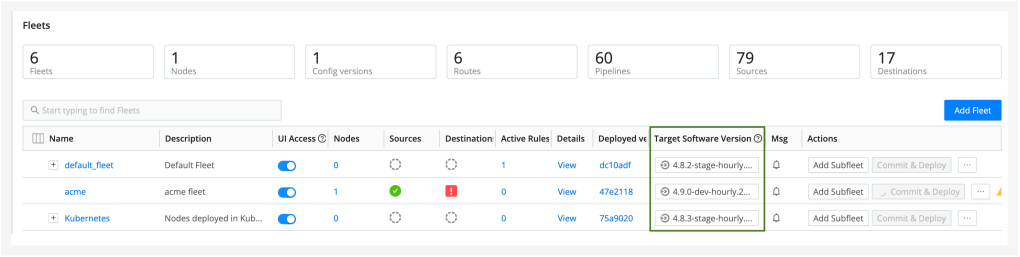
For details, see Fleet Upgrade Information.

10.1.6. Monitor Node Upgrade Status

Learn how to monitor the upgrading status via the UI.

To view the upgrade status of eligible Edge Nodes in a Fleet, select the **Target version** button.

You can find this button on the **Fleets** page (**Manage > Fleets**) or in the top bar of a selected Fleet.



Target version button on **Manage > Fleets**

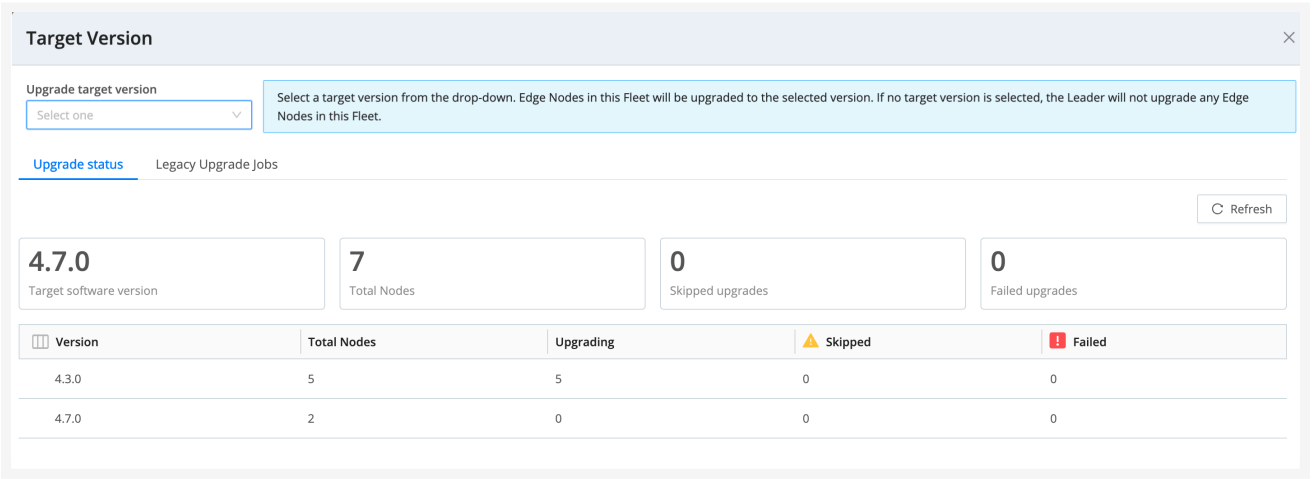


Target version button at the top of a selected Fleet

When you select this button, the **Target Version** window will open.

Explore the Target Version Window

The **Target Version** window provides information about Edge Node upgrades and allows you to select an upgrade target version for your Edge Fleets.



A popup window titled “Target Version” that shows the upgrade status of Edge Nodes in a Fleet

View the Status of Fleet Upgrades

The **Upgrade status** tab shows an overview of the Edge Nodes in this Fleet and the upgrade status for each Node version. The Nodes are organized by version, and the status columns display how many Nodes are in each status.

In this tab, you can view:

Status	Description
Version	Version for the Edge Nodes in this Fleet.
Total Nodes	Total number of Nodes in this Fleet running this version.
Upgrading	Nodes that are actively upgrading.

Skipped	Number of Nodes that were not upgraded due to one of these reasons:
	• The Fleet target version is not set.
	• No suitable upgrade package was found. You should use the **Path** package upgrade to add the specific upgrade package.
	• The Edge Node version is too old to receive an upgrade:
	  ◦ Version 4.3.0 and older for Windows Nodes
	  ◦ CentOS 6.x Nodes
	• The Leader couldn't download the upgrade package from the CDN.

Failed	The Node was not upgraded due to an error. The Node will retry the upgrade in the next hour.

View Upgrades for Legacy Edge Nodes

The **Legacy Upgrade Jobs** tab shows Leader-scheduled jobs that will upgrade Edge Nodes running versions 4.4.4 or older. If there are no legacy upgrade jobs, this tab will not show.

In this view, you can view the status the upgrade job per Edge Node.

- You can use the **Actions** icons to perform actions on the upgrade job: pause, stop, copy the link, delete, and view logs. If you cancel a job, you can view its artifacts.
- Click the **ID** to view job stats, settings, a live view, and logs.

10.1.7. Fleet Upgrading Use Case

An example upgrading use case, starring fictional solar energy company Greatgreen Goats.

Greatgreen Goats: A Fleet Upgrading Use Case

To help you conceptualize automatic Fleet upgrades, this use case walks you through how the fictitious company Greatgreen Goats upgrades Nodes in their two Fleets.

Greatgreen Goats is a small but rapidly growing company focused on installing and maintaining solar energy systems at goat farms around the country.

They use an on-prem Cribl Edge deployment to manage data on their Edge Nodes, divided into two distinct Fleets:

- **Fleet 1: Linux Operations.** This Fleet contains 20 Nodes, all running Edge 4.13.3. When the IT admin upgrades the Leader of Fleet 1 to 4.14.0, and they set the target upgrade version in the drop-down to **None**. Therefore, Nodes in Fleet 1 are never upgraded. They continue to remain at their current version, 4.13.3.
- **Fleet 2: Windows Monitoring.** This Fleet contains 15 Nodes. In this Fleet, the IT admin selects the current Leader version (in this case, 4.14.0) in the **Target upgrade version** drop-down. Now, when Nodes connect to the Leader, they pull down version 4.14.0 and upgrade to that version.

Fast forward four weeks: Cribl Edge has released a new version: 4.14.1. The IT admin upgrades the Leader to version 4.14.1.

Since Fleet 1 does not have a target version selected, Nodes in Fleet 1 remain at their current version, 4.13.3. If the IT admin decides to change the **Target upgrade version** to match the Leader in Fleet 1, the Nodes in Fleet 1 will be upgraded to 4.14.1.

Nodes in Fleet 2 will remain on version 4.14.0, which was the Leader's previous version, until the admin selects the next **Target upgrade version** (4.14.1, the Leader's new version). Then, Nodes will request the new version the next time they connect to the Leader and automatically upgrade.

This approach gives the IT admin at Greatgreen the flexibility to upgrade Nodes in Fleets independently from the Leader. Greatgreen can now scale and upgrade their Fleets with more reliability, since Nodes are pulling the release down from the Leader (instead of the Leader creating an overwhelming number of upgrade jobs).

10.1.8. Backup & Rollback Via The UI

Roll back a Cribl Edge upgrade using a backup.

When you initiate an upgrade through the UI, Edge first stores a backup of your current stable deployment. If the upgrade fails, then by default, Edge will automatically roll back to the stored backup package. You can adjust this behavior at **Settings** > **Global Settings** > **System** > **General Settings** > **Upgrade & Share Settings**, using the following controls.

Enable automatic rollback: Toggle on if Edge should automatically roll back an upgrade if the Edge server fails to start, or if the Edge Node fails to connect to the Leader. Toggle off to defeat this behavior.

Rollback timeout (ms): Time to wait, after an upgrade, before checking each Node's health to determine whether to roll back. Defaults to `30000` milliseconds, i.e., 30 seconds.

Rollback condition retries: Number of times to retry the health check before performing a rollback. Defaults to `5` attempts.

Check interval (ms): Time to wait between health-check retries. Defaults to `1000` milliseconds, i.e., 1 second.

Backups directory: Specify where to store backups. Defaults to `$CRIBL_HOME/state/backups`.

Backup persistence: A relative time expression specifying how long to keep backups after each upgrade. Defaults to `24h`.

10.1.9. Manual Rollback

Learn how to rollback a Linux Edge Node upgrade manually using a command line interface (CLI).

Manual Rollback

Using CLI commands, it's possible to explicitly roll back an on-prem Leader, and Nodes, to an earlier release. This works much like an upgrade.

Explicit rollback might be necessary if an automatic rollback fails. Otherwise, Cribl recommends first considering other options - an upgrade, or working with Cribl Support - before performing a manual rollback.

Rollback can encounter these complications:

- Your current configuration might take advantage of dependencies not supported in an earlier release.



Do not manually roll back any Cribl Edge instance running in a container. Instead, locate, download, and launch a container image hosting the earlier version you want.

Rollback Outline

We assume that you are rolling back to a previously deployed version that you know to be stable in your environment. The broad steps are:

1. Ideally, link your deployment to a Git remote repo, and commit and push your Leader's configuration to that remote. The repo will provide a stable location from which to restore your config, if necessary.
2. Stop the Leader instance. (From `$CRIBL_HOME/bin/`, execute `./cribl stop`.)
3. Also create a local backup of your Leader's whole `$CRIBL_HOME` directory.
4. Obtain the installation package for the earlier release and platform you need. (See the Rollback Example.)
5. Uncompress the earlier version to your original deployed directory. You can do this from the command line or programmatically (see the Rollback Example).

If installing to your existing target directory fails, try the same mitigations we list above for upgrades.

6. In a distributed deployment, Nodes must not run a higher version than the Leader. Repeat steps 2, 3, and 5 above on all Nodes, substituting "Node" for "Leader." (On Nodes where you've integrated Cribl Edge with systemd, stop the Cribl server with: `sudo systemctl stop cribl-edge`.)

Rollback Example

While rollbacks can be partially scripted, discovering earlier installation packages cannot be automated - the initial steps here are manual:

1. Stop and back up the Leader. (Follow steps 1-3 in the Rollback Outline.)
2. Open the **Cribl Releases** download page: https://cribl.io/download/.
3. In the **Cribl Past Releases** section, locate the older version that you want to restore.
4. Use the adjacent drop-down to select your target platform.
5. Right-click (or `Ctrl`-click) the corresponding **Download** button, and copy its URL to your clipboard. In this example, we've copied: `https://cdn.cribl.io/dl/4.13.0/cribl-4.13.0-46ac1714-linux-x64.tgz`
6. Swap that URL into an untar command like the following. Run this command from the parent (typically `/opt/`) of your existing `$CRIBL_HOME` directory:

```
sudo curl -Lso - https://cdn.cribl.io/dl/4.13.0/cribl-4.13.0-46ac1714-linux-x64.
```

Copy

7. Repeat the preceding steps to adjust all Nodes to a compatible version.

10.1.10. Troubleshoot Edge Upgrades

Find solutions to common issues you can encounter when upgrading Cribl Edge.

While we expect you'll have an easy time upgrading Cribl Edge, here is the troubleshooting information we've curated in case you do experience a hiccup:

Direct Upgrades

Cribl Edge does **not** support direct upgrades from any pre-GA version (such as a Cribl-provided test candidate) to a GA version. To get the GA version running, you must perform a new install.

Service Timeout for Leader Upgrades

If your Leader is managed by a service manager (such as systemd, openRC, or Upstart), the default `TimeoutSec=60` might be insufficient for complex environments.

The cumulative time required to parse dozens of Fleets, or environments with a high density of Sources, Destinations, and Routes, can easily exceed the 60-second window. If this threshold is hit, `systemd` will end the process mid-migration, causing the upgrade to fail.

To prevent this, temporarily increase the timeout:

1. Open the service configuration (or use an override): `systemctl edit cribl.service`
2. Set a higher limit, for example: `TimeoutSec=600`
3. Reload the configuration: `systemctl daemon-reload`
4. Retry the upgrade.

Once the upgrade completes, repeat these steps to revert to the default value, `TimeoutSec=60`. This keeps your stop/restart cycles fast by allowing `systemd` to intervene quickly if the process becomes unresponsive.

For more background on timeouts during larger data migrations, see details in Addressing HA Migration Timeouts.

Limited Feature Functionality After Leader Upgrade

Upgrading a Leader Node to a newer version may surface configuration options in the UI that will not be functional until you upgrade the connected Edge Nodes to the same version.

Cribl strongly recommends keeping Leader Nodes and Edge Nodes on the same version for the best experience and to ensure access to all features.

Error on Windows Edge Node Upgrades

When performing a Leader-initiated upgrade of Windows Edge Nodes in a Fleet, you can encounter an "Edge must be run as administrator to upgrade itself" error logged in `cribl.log`.

This error indicates that the service account under which the Cribl Edge service runs does not have proper permissions.

See Service Accounts and Edge Node Upgrades for information on how to configure service accounts to run Node upgrades in a Fleet.

Upgrade to Older Versions

Take a look at the following information if your Edge Nodes are running older software versions.

Upgrade to 4.2.x

Leaders on 4.2.x are compatible with Edge Nodes on 4.1.2, 4.1.3, and later. Due to a security update, Edge Nodes running on 4.0.4 cannot receive configurations from 4.2.x Leaders.

Upgrade to 4.1.x or Newer

Cribl Edge 4.1 and newer encrypts TLS certificate private key files when you add or modify them. See instructions for backing up your keys from earlier versions.

Upgrade to 4.0.x or Newer

Before upgrading your Leader to 4.0 and newer, see Persisting Socket Connections to prepare the host to keep communications open from Edge Nodes.

Safeguard Unencrypted Private Keys for Rollback

Cribl Edge 4.1 and newer encrypt TLS certificate private key files when you add or modify them.



Before upgrading from a version older than 4.1, make a backup copy of all unencrypted TLS certificate private key files. Having access to the unencrypted files is essential if you later find that you need to roll

back to your previous version.

To safeguard your unencrypted private keys, make a full backup of all Cribl config files. Files in the `auth/certs` directory are particularly important, such as those in:

- `groups/default/local/cribl/auth/certs/`
- `groups/<groupname>/local/cribl/auth/certs/`
- `cribl/local/cribl/auth/certs/`
- Etc.

Take appropriate precautions to prevent unauthorized access to these unencrypted private key files. If you need to roll back to a version older than 4.1, see Restoring Unencrypted Private Keys.

10.2. Notifications

With Notifications, you can set up alerts to notify Cribl Edge system administrators about events that require their immediate attention.

For example, you could set up a Notification to alert you when a Source stops sending data to a given Fleet, or when the volume of data sent to a Destination from the Fleet exceeds a threshold.



Notifications require certain plan or license tiers. Without an appropriate license, the configuration options described here will be hidden or disabled in Cribl Edge. For more information on license types, and a feature comparison, see Pricing.

How to Get Notifications

You can get notifications through:

- **Internal logs:** Cribl Edge stores these application-wide logs in the `notifications.log` file on the Leader Node, which also handles sending all Notifications. See Internal Logs for more information about logs.
- **Events in the UI:** Displays events in the Monitoring page. See Monitoring for more information about UI notifications.
- **Custom targets:** You can configure Notification targets to send alerts through email, pagers, Slack messages, and more. See Notification Targets for information about available targets and setup instructions.

Network connectivity for external targets: For any external Notification targets (email, PagerDuty, Slack, webhooks, AWS SNS, and similar), the Leader Node must be able to establish outbound connections to the corresponding service endpoints over the ports and protocols those services require (often HTTPS/443, or SMTP ports for email). On-prem deployments may need explicit firewall rules to allow this outbound traffic from the Leader.

Types of Notifications

Type	Available Notifications	For more info
Sources	• High Data Volume • Low Data Volume	Add a Notification for a Source

Type	Available Notifications	For more info
	• No Data • Source Persistent Queue Usage	
Destinations	• Destination Backpressure Activated • Destination Persistent Queue Usage • Unhealthy Destinations	Add a Notification for a Destination
Licenses	• Pending expiration of license	Add a Notification about License Expiration

You can also configure Cribl Stream to relay Notifications generated by other Cribl products. For an example, see the Cribl Search topic Notifications via Cribl Stream.

Notification Permissions

Notifications work with Cribl Edge role-based access control (RBAC). For users with non-administrative permissions, their assigned Roles and Policies determine the Fleets on which they can view Notification messages, configure Notifications, and configure targets.

Add a Notification for a Source

To create alerts for a Source or Collector:

1. Navigate to your Fleet. Then select **More** > **Sources**.
2. Select the Source type to display the table of its configured Sources. For example, select **Syslog** to set up notifications for a Syslog Source.
3. Select the **Notifications** button in the row for the Source you want to configure.
4. In the **Notification** modal, configure the following settings under **General**:
    - **ID**: Provide a unique ID for the Notification in this section.
    - **Enabled**: Toggle on (default) if you want Cribl Edge to enable the Notification. Toggle off to turn off the Notification.
5. In the **Configuration** section in the **When...** field, select the condition that will trigger the Notification and then configure the settings for that trigger condition. See Source Settings for more information about the trigger condition settings. The available Source Notification trigger conditions:
    - **High Data Volume**: Incoming data is above the data volume threshold for your configured time window.

- **Low Data Volume**: Incoming data is below the data volume threshold for your configured time window.
- **No Data Received**: The Source or Collector ingests zero data over your configured time window.
- **Persistent Queue Usage**: The Source persistent queue accumulates files past the threshold percentage of storage capacity. See Optimize Source Persistent Queues for more information about working with Source persistent queues to manage backpressure.

6. In the **Configuration** section, set the **Only notify on start and resolution** toggle. When toggled on, Cribl Edge will send a Notification when the triggering condition begins and then a second Notification to report when the condition has ended.
7. Configure the **Send Notification to** setting to set the Notification target(s). If you don't have a Notification target, the **Messages** icon will indicate you have a new Notification. See Notification Targets for more information about available targets and adding new targets.
   - Select **Add Target** to add an existing target.
   - Select **Create Target** to add a new target.
8. Save the Notification.

Your Notification will now appear on the **Events** tab within the configuration window for this Source. To view all Notification events and targets, select **Notifications** in the sidebar.

Consider testing the condition to determine if you configured the Notification properly by deliberately triggering the Notification trigger.

Source Notification Settings

The function of the **Data volume** and **Time window** settings change depending on which Notification trigger you selected:

Notification Trigger	Settings
High Data Volume	Triggers an alert if the total data ingested within the configured **Time window** exceeds the specified **Data volume** threshold.
Low Data Volume	Triggers an alert if the total data ingested within the configured **Time window** falls below the specified **Data volume** threshold.
No Data Received	Triggers an alert if a Source or Collector fails to ingest any data for the duration of the configured **Time window**.
Persistent Queue Usage	Triggers an alert when persistent queue disk usage exceeds the specified **Usage threshold**. The **Time window** determines how often this condition is checked.

Source name: This field locks to the Source for this Notification and cannot be edited.

Time window: This field's value sets the threshold period before the Notification will trigger. The default `60s` will generate a Notification when the Source has reported the trigger condition over the past 60 seconds. To enter alternative numeric values, append units of `s` for seconds, `m` for minutes, `h` for hours, and so forth.



See Source Notifications Best Practices for information about avoiding false notifications when configuring the **Time window** setting for Collectors that run on a schedule.

Data volume: Enter the threshold above which a Notification will trigger. Accepts numerals with units like `KB`, `MB`, and so forth. For example: `4GB`. If you want the unit to be bytes, enter the numeral only, without a unit designator.

Usage threshold: The percentage of full or used storage disk capacity. Cribl Edge generates a `Persistent Queue usage has surpassed <threshold>%` Notification when the Source persistent queue storage disk capacity is past the specified **Usage threshold** percentage. This field appears only when you configure a Notification for **Persistent Queue Usage**.

Metadata: User-defined fields included in the Notification payload. All Notification types can contain metadata. To add custom metadata fields, select **Add field** and add the fields as key-value pairs:

- **Name**: Enter a name for the custom field.
- **Value**: Enter a JavaScript expression that defines this field's value, enclosed in quotes or backticks. This can evaluate to a constant if needed.

Source Notification Best Practices

To ensure your Source Notifications are reliable and easy to manage, follow these best practices:

- For Collectors that run on a schedule, always set the **No Data Received** Notification time window to a duration that is longer than the time between scheduled runs. Because there is no data flow between runs, a shorter time window will trigger false alarms.
- When cloning a Source, remember to manually recreate any Notifications you need on the new Source. Cloning a Source does **not** copy its associated Notifications.

Add a Notification for a Destination

To create alerts for a Destination:

1. Navigate to your Fleet. Then select **More** > **Destinations**.
2. Select the Destination type to display the table of its configured Sources. For example, select **Syslog** to set up notifications for a Syslog Destination.
3. Select the **Notifications** button in the row for the Destination you want to configure.

4. In the **Notification** modal, configure the following settings under **General**:
    - **ID**: Provide a unique ID for the Notification in this section.
    - **Enabled**: Toggle on (default) if you want Cribl Edge to enable the Notification. Toggle off to turn off the Notification.
5. In the **Configuration** section in the **When...** field, select the condition that will trigger the Notification and then configure the settings for that trigger condition. See Destination Settings for more information about the trigger condition settings. The available Destination Notification trigger conditions:
    - **Destination Backpressure Activated**: The Destination has engaged its backpressure behavior, such as when a downstream Destination receiver is unreachable or slow to respond. This notification indicates that Cribl Edge detected blocked or dropped delivery during at least 5% of the trailing time window. It does not confirm that persistent queue has successfully engaged. The triggering conditions:
        - The Destination's **Backpressure behavior** is set to `Block` or `Drop`, and backpressure causes outgoing events to block or drop.
        - The Destination's **Backpressure behavior** is set to Persistent Queue and its **Queue-full behavior** is set to either `Block` or `Drop new data`. When the queue is full, it causes the Destination to block or drop outgoing events.
    - **Persistent Queue Usage**: The Destination persistent queue accumulates files past the threshold percentage of storage capacity. See Optimize Destination Persistent Queues for more information about working with Destination persistent queues to manage backpressure.
    - **Unhealthy Destination**: The health of the Destination has been in red status (as indicated on the Monitoring page) over the configured time window.
6. In the **Configuration** section, set the **Only notify on start and resolution** toggle. When toggled on, Cribl Edge will send a Notification when the triggering condition begins and then a second Notification to report when the condition has ended.
7. Configure the **Send Notification to** setting to set the Notification target(s). If you don't have a Notification target, the **Messages** icon will indicate you have a new Notification. See Notification Targets for more information about available targets and adding new targets.
    - Select **Add Target** to add an existing target.
    - Select **Create Target** to add a new target.
8. Save the Notification.

Your Notification will now appear on the **Events** tab within the configuration window for this Destination. To view all Notification events and targets, select **Notifications** in the sidebar.

Destination Notification Settings

The function of the **Time Window** settings change depending on which Notification trigger you selected:

Notification Trigger	Settings
Destination Backpressure Activated	The threshold for the Notification to trigger is: Cribl Edge detected a blocked or dropped state during ≥ 5% of the trailing **Time window**
Persistent Queue Usage	The **Time Window** determines how frequently the Notifications repeat.
Unhealthy Destination	The **Time Window** determines how frequently the Notifications repeat.

Destination name: This field locks to the Destination for this Notification.

Time window: This field's value sets the threshold period before the Notification will trigger. The default `60s` will generate a Notification when the Destination has reported the trigger condition over the past 60 seconds. To enter alternative numeric values, append units of `s` for seconds, `m` for minutes, `h` for hours, and so forth.

Usage threshold: The percentage of full or used storage disk capacity. Cribl Edge generates a `Persistent Queue usage has surpassed <threshold>%` Notification when the Destination persistent queue storage disk capacity is past the specified **Usage threshold** percentage. This field appears only when you configure a Notification for **Persistent Queue Usage**.

Metadata: User-defined fields included in the Notification payload. All Notification types can contain metadata. To add custom metadata fields, select **Add field** and add the fields as key-value pairs:

- **Name**: Enter a name for the custom field.
- **Value**: Enter a JavaScript expression that defines this field's value, enclosed in quotes or backticks. This can evaluate to a constant if needed.

Destination Notification Best Practices

When cloning a Destination, remember to manually recreate any Notifications you need on the new Destination. Cloning a Destination does **not** copy its associated Notifications.

Add a Notification About License Expiration

To create a license-expiration alert:

1. Navigate to **Settings** > **Global** > **Licensing**. Then select **Add Expiration Notification**.
2. In the **Notification** modal, configure the following settings under **General**:
    - **ID**: Provide a unique ID for the Notification in this section.
    - **Enabled**: Toggle on (default) if you want Cribl Edge to enable the Notification. Toggle off to turn off the Notification.

3. Optional: Add or create a Notification target. See Notification Targets for more information. If you don't have a Notification target, the **Messages** icon will indicate you have a new Notification.
4. Save the Notification.

To view all Notification events and targets, select **Notifications** in the sidebar. You can also reach this page by selecting **Settings** > **Global** > **Licensing**.

Consider testing the condition to determine if you configured the Notification properly by deliberately triggering the Notification trigger.

Edit Notifications

To edit a Notification:

1. Select **Notifications** in the sidebar. You can also reach this page by selecting an existing Notification in a Source or Destination modal. This page lists all Notifications.
2. Select anywhere on the row the Notification you want to edit.
3. To edit a Notification target (the delivery method for the Notification), use the **Targets** tab on this screen. See Notification Targets for additional details.

10.2.1. Email Notifications

If you're a Cribl Edge admin, email notifications make it easy to receive alerts about any operational issues that require your attention, such as a particular Source or Destination condition or a pending license expiration.

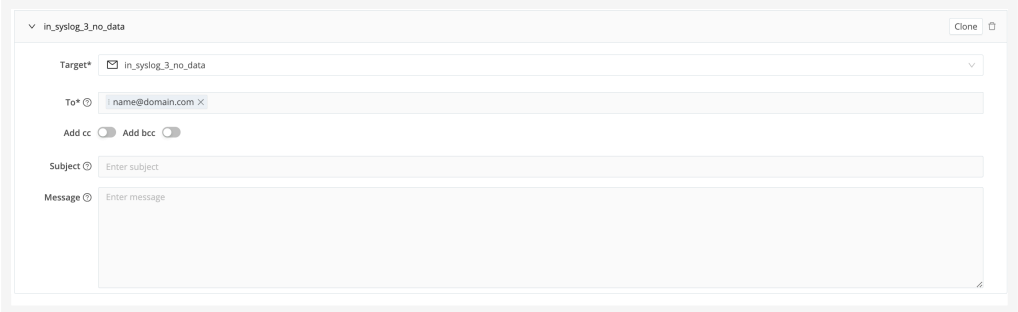
An email Notification requires two things – a configured Notification and an email Notification target. Cribl.Cloud Organizations using certain plan/license tiers also have access to a preconfigured email Notification target.

Email Notifications do not have an unsubscribe option. Recipients who do not want to receive particular email Notifications should contact their Cribl Edge admins.

Email Notifications are sent on a no-reply basis.

Configure Email Notifications

When you create a Notification for an email target, specify the recipients of the message, the subject line, and the contents of the message.



Email Notification with target selected

Target: The ID of the email target to which you want to send the Notification.

Select **Add target** to add an existing target.

Select **Create target** to create a new target. For information on configuring email Notification targets, see Email Notification Targets.

If the Notification already has a designated target, you can change the selection by selecting the drop-down.

When an email Notification target is selected, the following additional fields appear:

To: The email address of the recipient.

Add cc: When enabled, reveals a field where you can enter the addresses of additional recipients.

Add bcc: When enabled, reveals a field where you can enter the addresses of additional recipients that do not appear in the Notification email.

Cribl Edge does not limit the total number of recipients for a Notification, but your email service might set a limit.

Subject: The subject line of the email Notification. You can use variables in the subject line.

Message: The content of an email Notification. You can use variables in the body of the email message.

Email Notification Variables

Email variables are placeholders in the email template that get replaced with actual values when the email is sent. These variables can be:

- General-use variables, like `condition`, `worker_group`, or `timestamp`
- Special-purpose variables

You can use a variety of general-use and special-purpose variables in the subject or message body of an email Notification. Insert a variable name between two braces preceded by a `$`. For example: `${cribl_notification}`.

General-Use Email Variables

Variable	Description
`workspace`	Workspace name (Cribl.Cloud only).
`organization`	Organization ID (Cribl.Cloud only).
`timestamp`	Timestamp when the email is sent. For example: `2019-08-04 18:22:24 UTC`.
`cribl_notification`	User-defined notification ID.

Special-Purpose Email Variables

Variable	Usage	Notification type	Example
`_raw`	The `_raw` field of the Event triggering the Notification.	• License expiration • Cribl Search	`Your Cribl license expires in 3 days on 2019-08-04 18:22:24 UTC. Please update your license to continue`

Variable	Usage	Notification type	Example
			`processing data after 2019-08-04 18:22:24 UTC.`
`__worker_group`	Name of the Worker Group triggering the Notification.	• Destination Backpressure • High data volume • Low data volume • No data • Persistent Queue Usage • Unhealthy Destination	`default`
`timewindow`	Metric evaluation period used for Notification.	• Destination Backpressure • High data volume • Low data volume • No data • Persistent Queue Usage • Unhealthy Destination	`60s`
`status`	Status of the Notification Status of backpressure engagement.	• Destination Backpressure • High data volume • Low data volume • No data • Persistent Queue Usage • Unhealthy Destination	`OK`, `ALARM` `DISENGAGED`, `BLOCKING` or `DROPPING`
`input`	Type and name of a Source or Collector.	• High data volume • Low data volume • No data	`syslog:in_syslog_1`

Variable	Usage	Notification type	Example
`output`	Type and name of a Destination.	• Destination Backpressure • Persistent Queue Usage • Unhealthy Destination	`webhook:out_webhook_1`
`bytes`	Quantity of data triggering the specified Notification.	• High data volume • Low data volume • No data	`0`
`dataVolume`	Data volume which triggered the Notification. Accepts numerals with units like KB, MB, and so on.	• High data volume • Low data volume	`1G`
`health`	Health metric of the specified Source or Destination.	• Unhealthy Destination	`0`
`queue_usage`	Percentage of capacity set in the **Usage threshold** field.	• Persistent Queue Usage	`90`
`usage`	The actual usage, in percent.	• Persistent Queue Usage	`50`
`usageThreshold`	The usage threshold, in percent.	• Persistent Queue Usage	`50`

Troubleshoot Email Notifications

The section details the troubleshooting steps you can take if an email Notification fails to reach its intended recipient.

Test the Email Notification Target

An email Notification can fail if the target is misconfigured. You can test your email Notification target by following this procedure:

1. Open the target (in **Notifications** > **Targets**).
2. Select **Test Target**.
3. Add one or more email addresses to the **Test Target** modal and select **Send Test Email**.
4. Check the designated inbox to verify receipt of the test message. If it does not arrive in the designated inbox, review the target configuration.

Check Notification Service Logs

A failed email Notification leaves a log entry. You can examine logs by following this procedure:

1. Select **Monitoring** > **Logs**.
2. In the drop-down on the left, select **Leader** > **Notifications Service**.
3. Examine any logs that have errors.

Check Notifications

Cribl Edge stores Notifications. You can check them by following this procedure:

10.2.2. Notification Targets

A Notification target allows you to receive alerts using your preferred channel or alert delivery method. Every Notification requires one or more targets.

Available target types include:

- Webhook
- PagerDuty integration
- Slack
- AWS SNS
- Email
- System Messages (default)

When you create a new Notification, its default target type will be **System Messages**. When an event occurs in Cribl Edge to trigger a Notification condition, you will see an indicator in the **Messages** icon. Select the icon to open the **Messages** drawer and view the Notification details.

Add a Notification Target

To add a new Notification target:

1. In the Cribl Edge sidebar, select **Notifications**, and then select the **Targets** tab.
2. Select **Add Target** to open the **New Target** modal.
3. Give this target a unique **Target ID**.
4. Select the desired **Target type**.

Then configure the target as described in the appropriate page:

- Webhook
- PagerDuty
- Slack
- AWS SNS
- Email

Network requirement: Before you rely on a Notification target, confirm that the Leader Node can reach the target endpoint over the correct outbound ports (for example, HTTPS/443 for most SaaS integrations or SMTP ports for email gateways).

When you create a Notification target in Cribl Stream, Edge, or Search, it will also be available to you in the other products. For example, if you create a Notification target in Cribl Stream, you can also access it in Cribl Edge and Cribl Search.



Notifications require certain plan/license tiers. Without an appropriate license, target configuration options will be hidden or disabled in Cribl Edge's UI.

Manage Notification Targets

You can manage targets by selecting **Notifications** > **Targets**.

Select any existing Notification target to open a modal where you can manage the target, using buttons at the bottom of the page.

Select **Delete Target** to delete an existing Notification target. You can also select multiple targets for deletion, using the check boxes on the left side of the modal.

Select **Clone Target** to copy an existing target. A modal will open so you can modify the configuration of the target you started with.

To edit any target's definition in a JSON text editor, select **Manage as JSON** at the bottom of the **Notifications** > **Targets** modal. You can directly edit multiple values, and you can use the **Import** and **Export** buttons to copy and modify existing target configurations as `.json` files.

Some targets have additional controls for target configuration. See the documentation for the target of interest for more information.

10.2.2.1. Webhook Notification Targets

With this option, you can send Cribl Edge Notifications to an arbitrary webhook. Select **Webhook** in the **New Target** modal to expose multiple left tabs, with the following configuration options:

General Settings

Target ID: Enter a unique ID used to identify the target. This will show in the **Target ID** column of the **Targets** tab. You can't change it later, so make sure you like it.

Configuration

The added options that appear on this first left tab are:

URL: The endpoint that should receive Cribl Edge Notification events.



To proxy outbound HTTP/S requests, see System Proxy Configuration.

Method: Select the appropriate HTTP verb for requests: `POST` (the default), `PUT`, or `PATCH`.

Format: Specifies how to format Notification events before sending them to the endpoint. Select one of the following:

- `NDJSON` (newline-delimited JSON, the default).
- `JSON Array`.
- `Custom`, which exposes these additional fields:
    - **Source expression**: JavaScript expression whose evaluation shapes the event that Cribl Edge sends to the endpoint - for example: `notification=${_raw}`. For other fields you can use, see Expression Fields. If empty, Cribl Edge will send the full Notification event as stringified JSON.
    - **Drop when null**: Toggle on if you want to drop events where the above **Source expression** evaluates to `null`.
    - **Event delimiter**: Delimiter string to insert between individual events. Defaults to newline character (`\n`).
    - **Content type**: Defaults to `application/x-ndjson`. You can substitute a different content type for requests sent to the endpoint. This entry will be overridden by any content types set in this modal's **Advanced Settings** tab > **Extra HTTP Headers** section.

- **Batch expression**: Expression specifying how to format the payload for each batch. Defines a wrapper object in which to include the formatted events, such as a JSON document. This enables requests to APIs that require such objects. To reference the events to send, use the `${events}` variable. An example expression to send the batch inside a JSON object would be: `{"items" : [${events}] }`.

Expression Fields

When building the Source expression, you can use the following fields:

Fields Common to All Notification Types

- `starttime`: Beginning of the time bucket where this condition was reported. All Notifications have this field.
- `endtime`: End of the time bucket where this condition was reported. All Notifications have this field.
- `_time`: Timestamp when this Notification was created. All Notifications have this.
- `cribl_host`: Hostname of the (physical or virtual) machine on which this Notification was created. All Destination and Source Notifications have this.
- `cribl_notification`: Configured name/ID of this Notification. All Destination and Source Notifications have this.
- `origin_metadata`: Object containing metadata about the Notification's origin, with the following fields for all Destination Notifications:
  - `type`: "output".
  - `id`: ID of the affected Destination.
  - `subType`: Destination's type (where applicable).
- `origin_metadata`: Object containing metadata about the Notification's origin, with the following fields for all Source Notifications:
  - `type`: "input".
  - `id`: ID of the affected Source.
  - `subType`: Source's type (where applicable).

Unhealthy Destination

- `health`: Numeric value where `0`=green, `1`=yellow, `2`=red.
- `output`: Output ID of the affected Destination.

- `_raw`: "Destination `${output}` [in group `${__worker_group}`] is unhealthy".
- `_metric`: "health.outputs".

Destination Persistent Queue Usage

- `output`: Output ID of the affected Destination.
- `timeWindow`: The interval at which Notifications will be repeated.
- `usageThreshold`: The minimum persistent queue utilization (stated as a percentage) that will trigger Notifications.
- `_metric`: "system.pq_used".
- `usage`: The persistent queue utilization (stated as a percentage) that triggered the Notification.

Destination Backpressure Activated

- `backpressure_type`: `1` for Block, `2` for Drop.
- `output`: Output ID of the affected Destination.
- `_raw`: "Backpressure ([dropping|blocking]) is engaged for destination `${output}` [in group `${__worker_group}`]".
- `_metric`: "backpressure.outputs".

Source High Data Volume

- `health`: Numeric value where `0`=green, `1`=yellow, `2`=red.
- `bytes`: Number of bytes received in the time bucket.
- `input`: Input ID of the affected Source.
- `_raw`: "Source `${input}` [in group `${__worker_group}`] traffic volume greater than `${dataVolume}` in `${timeWindow}`".
- `_metric`: "total.in_bytes".

Source Low Data Volume

- `health`: Numeric value where `0`=green, `1`=yellow, `2`=red.
- `bytes`: Number of bytes received in the time bucket.

- `input`: Input ID of the affected Source.
- `_raw`: "Source `${input}` [in group `${__worker_group}`] traffic volume less than `${dataVolume}` in `${timeWindow}`".
- `_metric`: "total.in_bytes".

Source No Data Received

- `health`: Numeric value where `0`=green, `1`=yellow, `2`=red.
- `_time`: Timestamp when this Notification was created.
- `input`: Input ID of the affected Source.
- `_raw`: "Source `${input}` [in group `${__worker_group}`] had no data for `${timeWindow}`".
- `_metric`: "total.in_bytes"

License Expiration

- `severity`: One of "warn" or "fatal".
- `title`: One of: "License expiring soon, data will stop flowing." Or: "License has expired. Data flow has been stopped.".
- `text`: One of: "License will expire on `${expirationDate}`, no external inputs will be read" after that time. Please contact sales@cribl.io to renew your license." Or: "License has expired."

Authentication

Select one of the following options for authentication:

- **None**: Don't use authentication.
- **Auth token**: Use HTTP token authentication. In the resulting **Token** field, enter the bearer token that must be included in the HTTP authorization header.
- **Basic**: In the resulting **Username** and **Password** fields, enter HTTP Basic authentication credentials.

Post-Processing Settings

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Retries

Honor Retry-After header: Whether to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Edge limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. When enabled, any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. When disabled, all `Retry-After` headers are ignored.

Settings for failed HTTP requests: When you want to automatically retry requests that receive particular HTTP response status codes, use these settings to list those response codes.

For any HTTP response status codes that are not explicitly configured for retries, Cribl Edge applies the following rules:

Status Code	Action
Greater than or equal to `400` and less than or equal to `500`.	Drop the request.
Greater than `500`.	Retry the request.

Upon receiving a response code that's on the list, Cribl Edge first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Edge continues retrying the request without increasing the time between retries any further.

By default, this Destination has no response codes configured for automatic retries. For each response code you want to add to the list, select **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.

- **Backoff limit (ms)**: The maximum backoff interval Cribl Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: When you want to automatically retry requests that have timed out, toggle this control on to display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Reject certificates not authorized by a CA in the **CA certificate path**, nor by another trusted CA (for example, the system's CA). Defaults to toggled on.

Round-robin DNS: Toggle on to enable round-robin DNS lookup across multiple IP addresses, IPv4 and IPv6. When a DNS server resolves a Fully Qualified Domain Name (FQDN) to multiple IP addresses, Cribl Edge will sequentially use each address in the order they are returned by the DNS server for subsequent connection attempts.

Compress: Toggle on (default) to compress the payload body before sending.

Keep alive: By default, Cribl Edge sends `Keep-Alive` headers to the remote server and preserves the connection from the client side up to a maximum of 120 seconds. Toggle this off if you want Cribl Edge to close the connection immediately after sending a request.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests per Worker Process. When Cribl Edge hits this limit, it begins throttling traffic to the downstream service. Defaults to `5`. Minimum: `1`. Maximum: `32`.

Body size limit (KB): Maximum size of the request body before compression. Defaults to `4096` KB, but you can set this limit to as high as 500 MB (512000 KB).

High values can cause high memory usage per Edge Node, especially if a dynamically constructed URL causes this target to send events to more than one URL. The actual request body size might exceed the specified value because the target adds bytes when it writes to the downstream receiver. Therefore, we recommend that you experiment with the **Body size limit** value until downstream receivers reliably accept all events.

Events-per-request limit: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values can cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Extra HTTP headers: Name-value pairs to pass to all events as additional HTTP headers. Values will be sent encrypted. You can also add headers dynamically on a per-event basis in the `__headers` field; headers added by this method take precedence over headers defined in the **Extra HTTP headers** table.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Cribl Edge will redact all headers, except non-sensitive headers that you declare below in **Safe headers**.

Safe headers: Used to declare headers that are safe to log as plaintext. (Sensitive headers such as `authorization` will always be redacted, even if listed here.) Use a tab or hard return to separate header names.

10.2.2.2. PagerDuty Notification Targets

This option sends Cribl Edge Notifications to PagerDuty, a real-time incident response platform, using Cribl Edge's native integration with the PagerDuty API. Select Select **PagerDuty** in the **New Target** modal to expose the following additional options on the modal's (single) **General Settings** left tab:

General Settings

Target ID: Enter a unique ID used to identify the target. This will show in the **Target ID** column of the **Targets** tab. It can't be changed later, so make sure you like it.

Configuration

Routing key: Enter your 32-character Integration key on a PagerDuty service or global ruleset.

Group: Optionally, specify a PagerDuty default group to assign to Cribl Edge Notifications.

Class: Optionally, specify a PagerDuty default class to assign to Cribl Edge Notifications.

Component: Optionally, a PagerDuty default component value to assign to Cribl Edge Notification. (This field is prefilled with `logstream`.)

Severity: Set the default message severity for events sent to PagerDuty. Defaults to `info`; you can instead select `error`, `warning`, or `critical`. (Will be overridden by the `__severity` value, if set.)

Retries

Honor Retry-After header: Whether to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Edge limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. When enabled, any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. When disabled, all `Retry-After` headers are ignored.

Settings for failed HTTP requests: When you want to automatically retry requests that receive particular HTTP response status codes, use these settings to list those response codes.

For any HTTP response status codes that are not explicitly configured for retries, Cribl Edge applies the following rules:

Status Code	Action
Greater than or equal to `400` and less than or equal to `500`.	Drop the request.
Greater than `500`.	Retry the request.

Upon receiving a response code that's on the list, Cribl Edge first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Edge continues retrying the request without increasing the time between retries any further.

By default, this Destination has no response codes configured for automatic retries. For each response code you want to add to the list, click **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: When you want to automatically retry requests that have timed out, toggle this control on to display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

10.2.2.3. Slack Notification Targets

You can send Notifications to a Slack channel, using Slack's Incoming Webhooks.



You can also use a Webhook Destination to send specific event data to Slack for a team's attention. For more information, see our Slack/Webhook integration topic.

First, in your Slack workspace, use a Slack app to enable incoming webhooks.

Then, in the **New Target** modal, click **Slack** to expose the following additional options on the modal's (single) **General Settings** left tab:

General Settings

Target ID: Enter a unique ID used to identify the target. This will show in the **Target ID** column of the **Targets** tab. You can't change it later, so make sure you like it.

Configuration

Webhook URL: Add the full URL of your Slack Incoming Webhook. For example: `https://hooks.slack.com/services/T00000000/B00000000/XXXXXXXXXXXXXXXXXXXXXXXX`.

Retries

Honor Retry-After header: When toggled on and the `retry-after` header is present, Cribl Edge honors any `retry-after` header that specifies a delay, up to a maximum of 20 seconds. Cribl Edge always ignores `retry-after` headers that specify a delay longer than 20 seconds.

Cribl Edge will log a warning message with the delay value retrieved from the `retry-after` header (converted to ms).

When toggled off (default), Cribl Edge ignores all `retry-after` headers.

Settings for failed HTTP requests: Automatically retries after unsuccessful response status codes, such as `429` (Too Many Requests) or `503` (Service Unavailable). Clicking **Add Setting** reveals a table where you can add retry parameters for individual failed HTTP requests. These include:

- **HTTP status code**: The individual status code for which the retry parameters apply.

- **Pre-backoff interval (ms)**: How long, in milliseconds, Cribl Edge should wait before initiating backoff. The maximum interval is 600,000 ms (10 minutes).
- **Backoff multiplier**: Base for exponential backoff. A value of `2` (default) means Cribl Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so forth.
- **Backoff limit (ms)**: The maximum backoff interval, in milliseconds, Cribl Edge should apply. Default (and minimum) is 10,000 ms (10 seconds); maximum is 180,000 ms (180 seconds).

Retry timed-out HTTP requests: When toggled on, Cribl Edge to automatically retries HTTP requests that have timed out. Default is toggled off. Enabling this option exposes the following settings:

- **Pre-backoff interval (ms)**: How long, in milliseconds, Cribl Edge should wait before initiating backoff. Maximum interval is `600,000 ms` (10 minutes).
- **Backoff multiplier**: Base for exponential backoff. A value of `2` (default) means Cribl Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so forth.
- **Backoff limit (ms)**: The maximum backoff interval, in milliseconds, Cribl Edge should apply. Default (and minimum) is 10,000 ms (10 seconds); maximum is 180,000 ms (180 seconds).

10.2.2.4. AWS SNS Notification Targets

You can send Notifications to an Amazon Simple Notification Service (SNS) topic. This gives you access to a broad array of Notification destinations, such as various AWS services, mobile push notifications, or text messages.

To add an Amazon SNS Notification target in Cribl Edge, select **Notifications** in the sidebar, and then select **Targets** > **Add Target** > **AWS SNS**.

General Settings

Target ID: Enter a unique ID used to identify the target. This will show in the **Target ID** column of the **Targets** tab. It can't be changed later, so make sure you like it.

Configuration

Destination type: Defaults to **Topic ARN**. The SMS section explains the **Phone number** option.

Default Topic ARN: The default Amazon Resource Name (ARN) of the Amazon SNS topic to which you want to send Notifications. Enter the ARN in the following format:

```
arn:aws:sns:region:account-id:MyTopic

// for example:
// arn:aws:sns:us-west-2:000000000000:my-topic
```

Copy

If you use a non-AWS URL, the format must be:

```
{url}/myQueueName

// for example:
// https://host:port/myQueueName
```

Copy

The default topic ARN must be a JavaScript expression (which can evaluate to a constant value), enclosed in quotes or backticks. Can be evaluated only at initialization time. For example, if you're referencing a Global

Variable: `` `https://host:port/myQueue-${C.vars.myVar}` ``. This value can be overridden by the Notification event `__topicArn` field.

Topic type: The type of the topic selected in AWS SNS. Can be **Standard** or **FIFO**.

Message group ID: The tag that specifies that a message belongs to a specific message group. Messages that belong to the same message group are processed in a first in, first out (FIFO) manner. Must be a JavaScript expression (which can evaluate to a constant value), enclosed in quotes or backticks. Can be evaluated only at initialization time. For example, referencing a global variable: `` `https://host:port/myQueue-${C.vars.myVar}` ``.

Region: Select the Region associated with the Amazon S3 bucket.

Phone number allowlist: A wildcard list of phone numbers that are allowed to receive SMS notifications. This is used when **Destination type** is set to **Phone number**.

Authentication

Auto: This default option uses the AWS instance's metadata service to automatically obtain short-lived credentials from the IAM role attached to an EC2 instance, local credentials, sidecar, or other source. The attached IAM role grants Cribl access to authorized AWS resources. Can also use the environment variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`. Works only when running on AWS.

Manual: If not running on AWS, you can select this option to enter a static set of user-associated IAM credentials (your access key and secret key) directly or by reference. This is useful for Edge Nodes not in an AWS VPC, like those running a private cloud.

The **Manual** option exposes these additional fields:

- **Access key**: Enter your AWS access key. If not present, will fall back to the `env.AWS_ACCESS_KEY_ID` environment variable, or to the metadata endpoint for IAM role credentials.
- **Secret key**: Enter your AWS secret key. If not present, will fall back to the `env.AWS_SECRET_ACCESS_KEY` environment variable, or to the metadata endpoint for IAM credentials.

The values for **Access key** and **Secret key** can be a constant, or a JavaScript expression (such as `${C.env.MY_VAR}`) enclosed in quotes or backticks, which allows configuration with environment variables.

Secret: If not running on AWS, you can select this option to supply a stored secret that references an AWS access key and secret key. The Secret option exposes this additional field:

- **Secret key pair**: Use the drop-down to select an API key/secret key pair that you've configured in Cribl Edge's secrets manager. To store a new, reusable secret, select **Create**.

Assume Role

Enable for SNS: Toggle on to define an IAM Role to use, instead of automatically detecting one locally.

AssumeRole ARN: Enter the Amazon Resource Name (ARN) of the role to assume.

External ID: Enter the External ID to use when assuming role. This is required only when assuming a role that requires this ID to delegate third-party access. For details, see AWS' documentation.

Duration (seconds): Duration of the Assumed Role's session, in seconds. Minimum is 900 (15 minutes). Maximum is 43200 (12 hours). Defaults to 3600 (1 hour).

Post-Processing

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Advanced Settings

Maximum number of retries: The maximum number of retries before the output returns an error. The retries use an exponential backoff policy.

Endpoint: The SNS service endpoint. If empty, defaults to AWS' Region-specific endpoint. Otherwise, it must point to an SNS-compatible endpoint.

Signature version: Signature version to use for signing SNS requests. Defaults to `v4`.

Reuse connections: Whether to reuse connections between requests. Toggling on (default) can improve performance.

Reject unauthorized certificates: Whether to accept certificates that cannot be verified against a valid Certificate Authority (for example, self-signed certificates). Defaults to toggled on.

SMS Notifications

You can use an Amazon SNS Notification target to send text messages (SMS) to a list of phone numbers. To do this, you'll need to set up a allowlist of phone numbers that are permitted to receive Notifications.

1. Select **Notifications** in the sidebar, and then select **Targets** > **Add Target**.
2. Enter a unique **Target ID**.
3. Set the **Target type** to **AWS SNS**.
4. Set **Destination type** to **Phone number**.
5. Set **Region** to the Region of the Amazon S3 bucket.
6. In **Default Phone number**, enter a comma-separated list of phone numbers that are allowed to receive Notifications. This value can be overridden by the Notification event `__phoneNumber` field. You can use `*` as the wildcard character.
   For example: `+15555550123, +15555551***`.
7. If desired, use **Phone number allowlist** to specify a wildcard list of allowed phone numbers.
8. Configure the remaining sections of the AWS SNS Notification target as described earlier.
9. Select **Save**.

Now, when you set up Notifications, you can select the new Amazon SNS target and specify any phone number that matches the configured allowlist.

10.2.2.5. Email Notification Targets

You can send notifications by email, using an SMTP server of your choice. Once you have configured the email notification target, you can specify the recipients and customize the subject line and message content. See Email Notifications for details on configuring an email Notification.

To add an email Notification target in Cribl Edge, select **Notifications** in the sidebar, and then select **Targets** > **Add Target** > **Email**.

General Settings

Target ID: Enter a unique name to identify this email Notification target.

Configuration

Address: Identify the SMTP server by its hostname or IP address.

Port: Set the SMTP port. Use port `587` for SMTP Secure (SMTPS). Use `465` when SSL/TLS is enabled. You can also use port `2525` if your email service provider supports this port as a backup when other ports are blocked by a network provider or a firewall.

In Cribl.Cloud, Organizations on an Enterprise plan can use the default email notification target - `system_email` - and avoid configuring an SMTP server. If you **do** wish to configure an SMTP server in Cribl.Cloud, use any port other than port `25`.

Encryption type: Specify the encryption type used to secure SMTP communication. Options include:

- **STARTTLS**: Select this option to start the connection as plaintext, then upgrade it to a TLS-encrypted one if the server supports it. If the server doesn't support it, the connection remains plaintext.

  STARTTLS upgrades the connection to be encrypted but does not authenticate the server. Take additional steps to prevent man-in-the-middle attacks.

- **Require STARTTLS**: Select this option to require TLS. If the server doesn't support a secure connection, the connection will be dropped.
- **TLS (SMTPS)**: Select this option to use an encrypted connection from the start without requiring a subsequent connection upgrade.
- **None**: Select this option to use a plaintext connection.

Minimum TLS version: Optionally, select the minimum TLS version to use when connecting.

Maximum TLS version: Optionally, select the maximum TLS version to use when connecting.

Validate server certs: Toggle on to reject certificates that are **not** authorized by a CA in the **CA certificate path** or by another trusted CA (such as the system's CA).

From: Identify the email address of the sender.

Authentication

Username: The authentication principal (if required).

Password: The authentication credential (if required).

Test Email Notification Targets

When you finish configuring your email Notification target, you can manage it, using the following options.

Click **Save & Test Target** to save your target and open a modal where you can enter an email to test your email Notification target. After configuration, the button text changes to **Test Target**.

Default Email Notification Target for Cribl.Cloud Enterprise Orgs

For Cribl.Cloud Enterprise orgs, a default email Notification target is available for Cribl Stream, Edge, Search, and Insights. You can use this target to route any email Notification to any valid email address.

This target appears on the **Notifications** > **Target** page as `system_email`. You cannot modify or remove this target.

The `system_email` Notification target is managed by Cribl. It will be disabled in the unlikely event of abuse. If this target is disabled for a workspace, a log entry will appear in the Notifications Service logs. Select **Monitoring** > **Logs** > **Notifications Service** to view this log.



The `system_email` email Notification target is available only for Cribl.Cloud Enterprise organizations.

Send Domain for Cribl.Cloud Email Notifications

Every Cribl.Cloud workspace and organization has a unique address for its email Notification target. Messages sent using this target will have a sender's address in the form `do-not-reply@<workspaceName>-`

`<organizationId>.criblcloud.email`. To ensure delivery, recipients should add the `criblcloud.email` domain to their email allowlists.

10.3. Uninstall Cribl Edge From Linux

Removing a Cribl Edge installation from Linux, whether for a clean reinstall or permanently.

Uninstalling a Single Edge Node

- (Optional) To prevent systemd from trying to start Cribl Edge at boot time, run the following command: `sudo $CRIBL_HOME/bin/cribl boot-start disable`
  If you're running both Cribl Edge and Cribl Stream on the same host, be sure to execute this command in the same mode (Edge or Stream) and the same directory in which you originally ran `cribl boot-start enable`.
- Stop Cribl Edge (stopping the main process).
- Back up any necessary configurations/data.
- Delete the `cribl` user: `userdel cribl`
- Delete the user group: `groupdel cribl`
- Remove the directory where Cribl Edge is installed.
- Directly delete any local files that Cribl Edge added while running, such as the `.dat` file and other local configs.
- Remove any leftover files in the system's temporary directory. On Linux, this is `/tmp` by default.
- Remove any related configuration files and scripts, and disable the Cribl service if using a service manager.

In a Distributed deployment, repeat the above steps for the Leader instance and all Edge Node instances.

Uninstalling a Distributed Deployment

In a distributed deployment, repeat the above steps on:

- The Leader instance.
- All Edge Node instances.

10.4. Uninstall Cribl Edge From Windows

Cleanly remove a Cribl Edge instance from Windows.

You can uninstall Cribl Edge - whether for a clean reinstall or permanently - either via the Windows UI or via your command prompt.

Using the Windows UI

To uninstall Cribl Edge using Windows' graphical UI:

1. On your Windows Server, search for **Add or remove programs**.
2. In the **Apps & features** list, scroll to **Cribl Edge**, and click **Uninstall**.
3. In the confirmation dialog, click **Yes**.

A success message will confirm removal.



This uninstall process does not automatically delete the `local`, `log`, `pid`, and `state` directories, because those folders include contents generated after the original installation completed. To completely remove the application, you must explicitly delete the folders from the filesystem.

Using the Command Prompt

If you installed Cribl Edge using the Msi Installer, you can uninstall it by running the following at a command prompt:

```
- msiexec /x cribl-<version>-<build>-<arch>.msi
```

Copy

10.5. Uninstall Cribl Edge From MacOS

Cleanly remove a Cribl Edge instance from macOS.

To cleanly remove a Cribl Edge instance from macOS:

1. Stop the service:

```
sudo launchctl unload /Library/LaunchDaemons/io.cribl.plist
```

Copy

2. Remove the installed Edge Node:

```
sudo rm -rf /Library/LaunchDaemons/io.cribl.plist /opt/cribl
sudo pkgutil --forget io.cribl.pkg.edge
```

Copy

10.6. Customize The Interface

You can customize the Cribl Edge interface to adapt it to your needs by adding:

- A custom login page with a selected logo and additional explanatory text.
- Custom banners to serve information, warnings, or news to users.

10.6.1. Custom Banners

Cribl admins can define a custom banner from the Leader. This will be displayed at the top of your Cribl Organization, above all Cribl apps. You can display one banner at a time.

Your banner can display warnings, requirements, neutral updates, or good news about new features or options. It can contain a mixture of text, a raw URL, or a URL with a friendly/short label. The banner does not appear unless you enable and configure it, as described here.

1. In the sidebar, select **Settings** > **Global**.
2. Select **System** > **Custom Banner**.
3. Toggle **Enable banner** on to expose the configuration options.
4. From the **Background color** dropdown, select a background color for your banner. You can also pick a custom color by selecting **+Add color** in the dropdown.
5. Optionally, toggle **Invert front color** to **Yes** to make the text on the banner white instead of black (typically, for darker custom banner colors).
6. In the **Banner message** field, enter a single-line message of up to 100 characters. The displayed banner will truncate characters that cannot fit across one line of a given viewport.
    - You can use emojis and special characters in the banner message. To open the menu of supported options, place the cursor in the **Banner message** field and press **Ctrl+Cmd+Space** (macOS) or **Windows key + semicolon** (Windows).
7. In the **Link URL** field, optionally enter a hyperlink to append to the message.
8. In the **Link display** field, optionally give your **Link URL** a text label. This can include up to 100 characters, but a long label can force-truncate the **Banner message**. A short label helps everything fit.
9. Select **Save** to display your banner.
10. Commit your changes from the Leader.

To hide the banner, toggle the **Enable banner** off, and again **Save** and commit your changes. Your banner's configuration will remain intact, ready to re-enable and/or revise when you're ready.

10.6.2. Custom Login Page

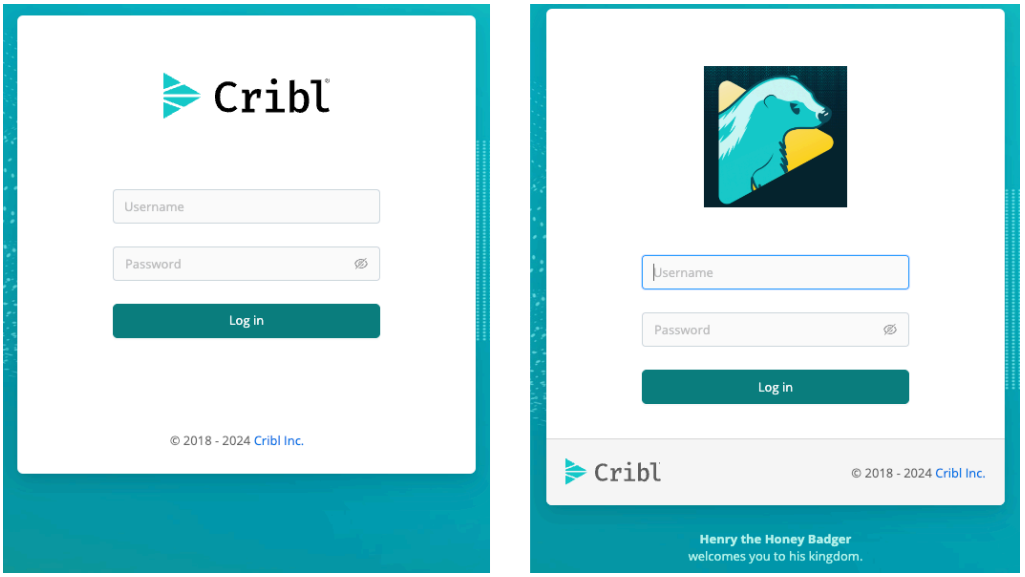
You can customize the look of the login page for all Cribl apps.

Custom login pages are available only in on-prem installations.

Configure Custom Login Page

To configure a custom login page:

1. In the Cribl Edge sidebar, select **Settings**, then **Global**.
2. Under **System**, select **General settings**.
3. Go to **Custom login page** and toggle **Enabled** on.
4. In **Logo file**, select an image file that will serve as a logo. The image must be no larger than 350x350px, under 2MB, and in .png, .jpg, or .jpeg format.
5. In **Content**, edit the provided HTML to modify the text block displayed under the login box.



Built-in and customized login pages

10.7. Scripts

Scripts are available only in on-prem deployments, not in Cribl.Cloud.

Admins can run scripts (for example, shell scripts) from within Cribl Edge by configuring and executing them at **Settings** > **Global** > **Scripts**. Scripts are typically used to call custom automation jobs or, more generally, to trigger tasks on demand. For example, you can use Scripts to run an Ansible job, or to place a call to another automation system, when Cribl Stream configs are updated.

With Great Power Comes Great Responsibility!

Scripts will allow you to execute almost anything on the system where Cribl Edge is running. Make sure you understand the impact of what you're executing before you do so!

Enable Scripts

For security reasons, using scripts is disabled by default on all new deployments. You can re-enable them by using the `limits` command (available since Cribl Edge 4.8.2).

To control script availability, run:

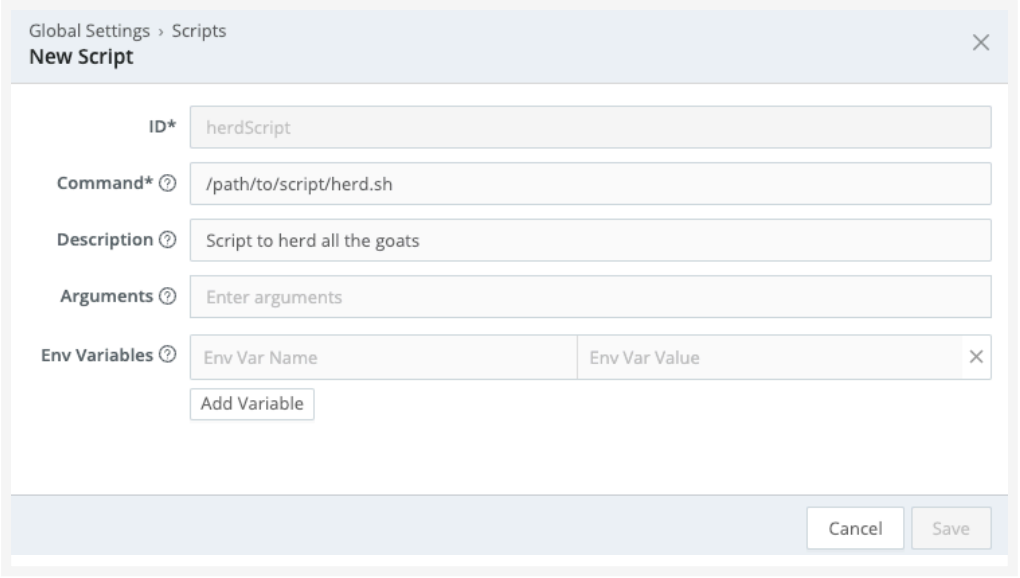
```
# Enable scripts
./cribl limits -s true
# Disable scripts
./cribl limits -s false
```

Copy

When you enable or disable scripts, you need to restart the instance for the changes to take effect.

Add Scripts

To add a new script, select **Add Script** on the scripts settings page.



Settings > Scripts page

The **New Script** modal provides the following fields:

- **ID**: Unique ID for this script.
- **Command**: Command to execute for this script.
- **Description**: Brief description about this script. Optional.
- **Arguments**: Arguments to pass when executing this script.
- **Env variables**: Extra environment variables to set when executing script.

Scripts in Distributed Deployments

- Scripts can be deployed from the Leader Node, but can be **run** only locally from each Worker Node.
- If the Script command is referencing a file (for example, `herd.sh`), that file must exist on the Cribl Stream instance. In other words, the Script management interface cannot be used to upload or manage script files.

11. Secure Your Deployment

Secure your Cribl Edge deployment, whether hosted on Cribl.Cloud, hybrid or on-prem, by following these guides:

- Secure your Cribl.Cloud Deployment
- Cribl.Cloud Security Practices
- Secure your On-Prem/Hybrid Deployment
- Secure Leader and Communication
- Manage Secrets and Keys
- Compliance

11.1. Secure Your Cribl.Cloud Deployment

Cribl.Cloud is SOC 2 Type II-certified and GDPR-compliant, simplifying compliance with industry standards and regulatory requirements. For Cribl.Cloud security architecture details, see Cribl.Cloud Security Practices.

For formal compliance reports and penetration test executive summaries, see Request Access to TrustCloud.

To secure your Cribl.Cloud deployment, consider implementing the following.

Secure Your Cribl.Cloud Account

Create a strong password during your initial setup.

To create a secure password, ensure it is at least 12 characters long and includes a combination of lowercase letters, uppercase letters, numbers, and special characters (!@#$%^&*). Follow the on-screen instructions carefully to meet these requirements.

(For Cribl.Cloud setup instructions, see Initial Cribl.Cloud Setup).

Review Your Network Settings

When you log into your Cribl.Cloud Organization, navigate to the **Workspace** link. Here you can check and manage connectivity details - Data Sources, Access Control, and Trust relationships - for your Cribl-managed Workers in Cribl.Cloud.

On the **Data Sources** tab, you can review the available ports and TLS configurations.

Note that TLS encryption is pre-enabled for you on some, but not all Sources. For more information, see TLS in Cribl.Cloud.

Configure TLS Mutual Authentication on Cribl.Cloud

Within Cribl.Cloud, enabling mutual TLS authentication for individual data sources requires a trusted Certificate Authority (CA) certificate chain. This CA certificate verifies the legitimacy of the client certificate presented by the source during connection. For details, see TLS Mutual Authentication in Cribl.Cloud.

Set Up Role-Based Access Control for Members

Cribl.Cloud offers detailed role-based access control (RBAC) to ensure appropriate permissions and security. For details on how to invite and manage the roles for Members of your Organization, see Members.

Set Up Your Preferred Authentication Method

When your Cribl.Cloud Organization uses a certain plan, you have a range of secure and convenient login options to fit your Organization's needs, including:

- Local Accounts - Create traditional usernames and passwords for direct login. For details, see Local Authentication.
- Federated Authentication (OIDC and SAML): Integrate with existing identity providers (IdPs) like Okta or Entra ID for centralized user management and single sign-on (SSO). For details, see Cribl.Cloud SSO Setup.

Data at Rest

Cribl.Cloud provides built-in encryption for data at rest, ensuring that your information remains secure within its infrastructure.

Secure Data in Motion

For an extra layer of protection, you can enrypt individual fields within events. For details, see Encryption of Data in Motion.

Currently, Cribl Edge supports decryption only when Splunk is the end system. For details, see Decryption of Data in Splunk.

For a comprehensive solution, we recommend configuring TLS. TLS encrypts the entire data stream, including all event fields, regardless of Source or Destination type. This provides a more robust security posture for your data in motion. For details, see Secure Cribl Edge Sources and Destinations with Certificates.

Configure a Key Management Service (KMS)

A Cribl.Cloud Organization on a certain plan offers a Key Management Service (KMS) that strengthens the security of your data. KMS securely manages the encryption keys used to protect sensitive information on Worker Groups and Nodes within Cribl. For details on how to activate and configure KMS, see KMS Configuration.

(Optional) Segment Your Organization with Workspaces

Segment your organization's data and user access using Workspaces. This multi-tenancy feature allows you to create isolated instances within your Cribl.Cloud Organization, addressing critical security, compliance, and data isolation requirements tailored for diverse business units.

Each Workspace boasts a dedicated virtual private cloud (VPC). This essentially creates a separate environment within your Organization for each Workspace, ensuring the complete isolation of data and users in different Workspaces.

For details, see Configuring Workspaces.

(Optional) Create Isolated Data Environments Using Stream Projects

Within Cribl Stream, Projects offer a more granular level of isolation allowing teams and users to manage and share data flows without impacting others. Projects enable you to define the data that the Project's users can consume, where that data can be sent, and who has access to the Project.

Projects are ideal for teams and users who need to share and manage specific data, with accelerated access to relevant data and minimal configuration requirements. Data can be forked into full-fidelity versus redacted views to match the authorzation requirements and needs of different teams or users.

For details, see Configuring Projects.

11.2. Secure Your On-Prem/Hybrid Deployment

This document outlines essential security considerations to safeguard your on-prem/hybrid Cribl deployment.

Secure Your Installation

Secure your Cribl Edge deployment by ensuring the following during installation.

Choose a Secure Environment

Protect your Cribl deployment by:

- **Isolating your deployment** - deploy Cribl Edge on a dedicated network segment to minimize exposure.
- **Restricting physical access** - control who can physically access the deployment hardware.

Configure a System Proxy

Enhance the security and reliability of your Cribl deployment by leveraging proxy servers.

- To configure a system proxy for outbound HTTP/S traffic: Set the HTTP_PROXY and HTTPS_PROXY environment variables before starting Cribl Edge, specifying the proxy's IP address or FQDN and optionally the port number. For details, see System Proxy Configuration.
- To configure a SOCKS proxy for secure communication between Leader and managed Nodes: Follow the steps in Configure a SOCKS proxy. This will allow you to bypass network restrictions and ensure uninterrupted data flow.

Create a Dedicated User

To enhance security, avoid running Cribl Edge with root privileges. Instead, create a dedicated user account with limited permissions.

For details, see Setting Capabilities for Cribl Stream or Running Edge as an Unprivileged User.

Change Default Admin Password

After your initial login, immediately create a strong, unique password for the administrator account. This will help prevent unauthorized access.

Note: Edge Node passwords are managed by the Leader Node.

Security Files

For details on the `cribl.secret` master encryption key and other security files, see How Secrets Are Protected.

You can use the **Settings** > **Security** > **Secrets** section to create and update authorization tokens, username/password combinations, and API-key/secret-key combinations for reuse across the application. For details, see Secrets.

Set Your Remote Git Repo to Private

Cribl uses Git for efficient configuration management. In Distributed deployments, a local Git repository tracks configuration changes. For added security and disaster recovery, you can also connect to a remote Git repository. Here are some key considerations.

- **Private repositories** - Always ensure that your remote Git repository is private, to prevent unauthorized access to your configuration files.
- **Disaster recovery** - Remote repositories provide a valuable backup for your Cribl configuration in case of system failures or data loss.

FIPS Mode (Cribl Stream only)

Do not start Cribl Stream without FIPS mode enabled, or else you will be unable to run it in FIPS mode later. You must enable FIPS mode as described in the FIPS Mode document, after installing **but before starting** Cribl Stream.

Secure the Leader Auth Token

Cribl Edge generates a strong, random authentication token for new installations and when transitioning from Single-instance to Distributed mode.

If you need to change the default token:

- Create a unique, secure value.
- Update the token in the UI or Docker Compose file.

For details, see How to Secure the Auth Token for the Leader Node.

Set up Role-Based Access Control for Members

Cribl offers detailed role-based access control (RBAC) to ensure appropriate permissions and security. For details on how to define and manage fine-grained access control across Cribl products and resources, see Members and Permissions.

Enhance Security with an Identity Provider (IDP)

We strongly recommend using an external Identity Provider (IDP) to implement single sign-on (SSO) for Cribl. This enhances your organization's security, by centralizing user authentication and authorization.

For details, see:

- SSO with Okta and OIDC (on-prem)
- SSO with Okta and SAML
- SSO with Microsoft Entra ID (On-Prem)

Secure Your Communication

Secure your Cribl communication channels by using the following options.

- **SSL/TLS** - Encrypt Cribl access and traffic with SSL or TLS.
- **Custom headers** - Enhance security with custom HTTP headers.
- **Encryption** - Create and manage your own keys to encrypt events in real time.
- **Secrets** - Centrally manage secrets that your instances use to authenticate on integrated services.
- **KMS** - Use internal or external key management service for advanced key management.

Choose a combination that best aligns with your organization's security policies.

Secure Access to the Leader and Edge Nodes

You can secure access to the Cribl Edge API and UI by configuring Secure Sockets Layer (SSL) or Transport Layer Security (TLS). Do this on the Leader, to secure inbound communications to the Edge Nodes. You can use your own certs and private keys, or you can generate a pair with OpenSSL. For details, see Configure TLS for API and UI Access.

For TLS details, see the following:

- TLS version support and defaults
- TLS settings and Traffic Types
- System-wide TLS Settings Including Ciphers

Import an Existing Certificate and Key

You can secure Leader-to-Edge Node communications by using an existing TLS/SSL certificate and key, in `\.pem\` format. For details, see Import Certificate and Key.

Create a Self-signed Certificate

You also have the option to create a self-signed certificate and key by following the steps outlined in Configure TLS for API and UI Access.

Connect to the Leader Securely

Configure secure communication between your Leader and Edge Nodes using either the user interface, the `instance.yml` configuration file, or environment variables. For details, see Secure Leader/Edge Nodes Communications.

Secure Access to the Worker/Edge Nodes UI

Cribl recommends that in Enterprise Distributed deployments, you prevent direct browser access to the UI for all Edge Nodes. For details, see Secure Access to Edge Node UIs.

Configure TLS Mutual Authentication (optional)

Implement client certificate exchange to enhance security by allowing only authorized clients with valid certificates to connect to the Leader. Configure this feature after setting up encrypted TLS communication. For details, see Configuring TLS Mutual Authentication.

Secure Communication Between Sources and Destinations

To authenticate Cribl Edge with external data senders and receivers, configure certificates at the Fleet level. For details, see Secure Cribl Edge Sources and Destinations with Certificates.

Configure Custom HTTP Headers

Custom HTTP headers are metadata that can be sent along with HTTP requests and responses. They provide a way to convey extra data that is not part of the standard HTTP protocol. You can customize Cribl's security settings with custom HTTP headers. For details, see Custom HTTP Headers.

Create Encryption Keys

You can create and manage keys that Cribl Edge will use for real-time encryption of fields and patterns within events. For details, see:

- Accessing Keys.
- Add New Key.
- Get Key Bundle.
- Safeguarding Unencrypted Private Key Files for Rollback (Edge/Stream).
- Restoring Unencrypted Private Keys.

Configure a Key Management Service (KMS)

Cribl offers a built-in Key Management Service (KMS) that strengthens the security of your data. With Standard and Enterprise license tiers, you have the option to integrate an external KMS. A KMS securely manages the encryption keys used to protect sensitive information on Fleets and Edge Nodes within Cribl Edge.

For details, see Configure KMS.

Create Secrets

Use the Cribl secrets store to isolate and simplify the management of authentication tokens, usernames/passwords, and API keys.

- To understand the master encryption key, see How Secrets Are Protected.
- To access secrets, see Accessing Secrets.
- To add a new secret, see Add New Secret.
- For secret types, see Secret Type.

Secure Data at Rest

You are solely responsible for implementing encryption for data at rest on systems hosting Cribl products. To secure data at rest, consider:

- **Filesystem-level encryption**: Encrypting the entire file system where Cribl data resides.
- **Hardware-level encryption**: Using self-encrypting drives (SEDs) or other hardware-based encryption mechanisms.

Secure Data in Motion

For an extra layer of protection, consider encrypting your data while it travels between Cribl Edge and your data Sources and Destinations. For details, see Data Encryption.

Currently, Cribl Edge supports decryption only when Splunk is the end system. For details, see Data Decryption.

For a comprehensive solution, we recommend configuring TLS. TLS encrypts the entire data stream, including all event fields, regardless of Source or Destination type. This provides a more robust security posture for your data in motion. For details, see Secure Cribl Edge Sources and Destinations with Certificates.

Disable the Exec Source

The Exec Source, while powerful, can pose security risks if not configured and used carefully. To mitigate these risks, you can disable the Exec source by setting the `CRIBL_NOEXEC` environment variable. This prevents the execution of arbitrary commands, reducing the attack surface. For details, see Disable the Exec Source.

Filter Sensitive Information in API Responses

Use the `sensitiveFields` configuration property to filter sensitive information in the responses to API requests to `/inputs` and `/outputs` endpoints. Add an array that lists the attributes you want to filter under `sensitiveFields` in the `api` section of your `cribl.yml` configuration file, as shown in this example:

Example sensitiveFields array in cribl.yml

```
api:
  host: 0.0.0.0
  port: 9000
  sensitiveFields:
    - awsApiKey
    - awsSecret
    - token
    - password
  retryCount: 120
  ...
```

Copy

In the responses for `/inputs` and `/outputs` endpoints, Cribl returns empty values (`''`) for the attributes you list in the `sensitiveFields` array. In this example, Cribl will return empty values for the attributes `awsApiKey`, `awsSecret`, `token`, and `password`.



Use caution: the `sensitiveFields` configuration property affects UI workflows and applications that rely on the responses to `/inputs` and `/outputs` endpoints. The responses will contain empty values (`''`) instead of plaintext values for the attributes that you list in the `sensitiveFields` array. When you change or update a Source or Destination, you must also re-enter the correct value for any attribute that is listed in the `sensitiveFields` array.

Refer to the API reference for more information about the `/inputs` and `/outputs` endpoints and the attributes that appear in their responses.

11.3. Secure Leader/Edge Nodes Communication

You can configure secure communication between your Leader and Edge Nodes. As an additional security measure, we recommend you prevent direct browser access to the UI for all Edge Nodes.

Disable Exposing the API Service for Edge Nodes

Cribl recommends that in enterprise distributed deployments, you disable exposing the API service to the network on the configured API port for all Edge Nodes.

1. Select a Fleet.
2. Open **Fleet Settings** (top right).
3. Under **General Settings** > **API Server Settings**, select **Advanced**.
4. Toggle **Listen on port** off.
5. Select **Save**.

Secure Browser Access to Edge Node

Cribl recommends that in enterprise distributed deployments, you prevent direct browser access to the UI for all Edge Nodes.



On the Leader Node, start by enabling UI access to Edge Nodes (Stream, Edge). This way, admins will still be able to tunnel through from the Leader to the UI on any Edge Node. This is also a prerequisite for Connect to the Leader Securely described below.

1. Select a Fleet.
2. Open **Fleet Settings** (top right).
3. Under **General Settings** > **API Server Settings**, select **Advanced**.
4. Toggle **Local UI access** off.
5. Select **Save**.

Connect to the Leader Securely

For Cribl.Cloud Leader deployments, TLS is enabled by default, so you don't need to configure it manually – the Cribl.Cloud platform handles this automatically. The TLS configurations described below

> are only for on-prem and self-managed deployments.

You can configure secure communication between your Leader and Edge Nodes using the UI, the `instance.yml` config file, or environment variables.

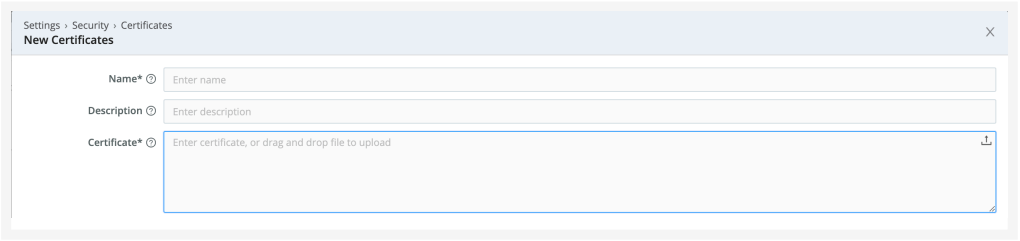
Secure via the UI

To set up secure communication via the UI, you configure first the Fleet, then the Edge Node, then the Leader.

Fleet Setup

For each Fleet whose Edge Nodes you want to secure:

1. Open your TLS/SSL certificate file, and copy its contents to your clipboard. (This can be the same certificate you uploaded to the Leader, or a different certificate.)
2. In the sidebar, select **Fleets**, and then select the Fleet you want to configure.
3. Select **Group Settings** or **Fleet Settings** (upper right).
4. Select **Security** > **Certificates** > **TLS**.
5. Click **Add Certificate**.
6. In the **New Certificates** modal, add the certificate's contents to the **Certificate** field.
   (You can paste the file's contents, or you can drag/drop or upload the `.pem` file.)
7. As you did on the Leader, also insert your **Private key**, and (as needed) your **Passphrase** and **CA certificate**.
8. Click **Save**.
9. Commit and Deploy the Fleet's new configuration, including the new certificate.
10. Repeat the preceding steps on each Fleet.

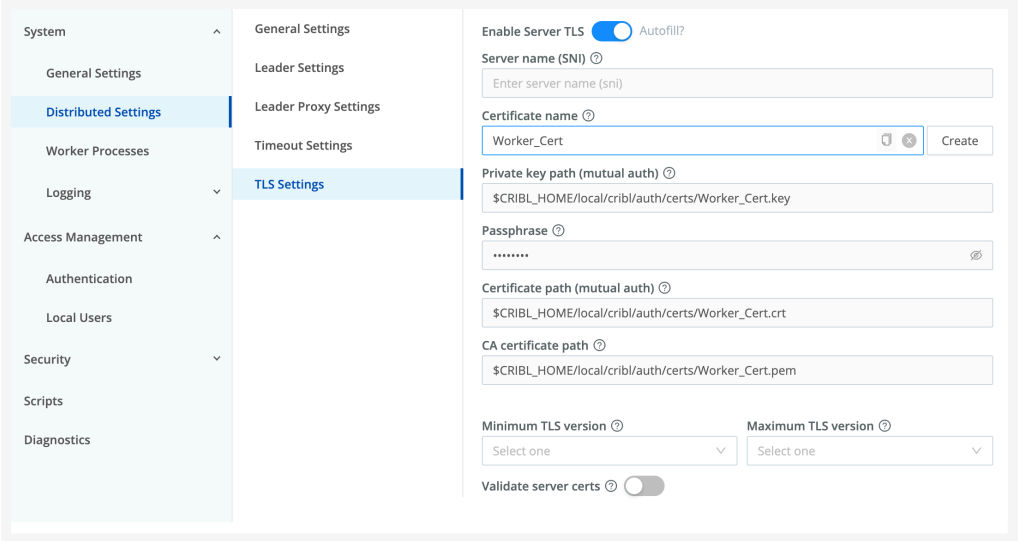


Fleet-level certificates are configured like the Leader's certificate

Edge Node Setup

For each Edge Node that you want to secure:

1. Enable the Leader's UI access to each desired Edge Node (Stream, Edge).
2. Tunnel through from the Leader to the Edge Node's UI.
3. Navigate to this Edge Node's **Worker Settings** (upper right) > **System** > **Distributed Settings** > **TLS Settings**.
4. Toggle **Enable Server TLS** on.
   This will expose the remaining TLS settings.
5. From the **Certificate name** drop-down, select the certificate you uploaded to the parent Fleet.
   This will prefill all the required fields. (See all deployed certificates at the left nav's **Security** > **Certificates** link.)
6. Click **Save**.
   The Worker will be unavailable during a short lag, while it restarts with the new configuration.
7. Repeat the preceding steps on each Edge Node.



Configuring TLS on Edge Node's UI, from the Leader

Leader Setup

Next, return to the Leader's UI:

1. Select **Settings** > **Global** > **System** > **Distributed Settings** > **TLS Settings**.
2. Toggle **Enable server TLS** on.
3. In the **Certificate name** drop-down, select an existing certificate. This will auto-populate the corresponding certificate fields.
4. Click **Save**.

After you've enabled TLS on the Leader, generating bootstrap scripts to add or update Edge Nodes will automatically use the `tls://` protocol for the Leader's URL.

Secure via the YAML Config File

You can also configure the Leader's `$CRIBL_HOME/local/_system/instance.yml` file to ensure that TLS is enabled. Here's the relevant section:

```
distributed:
  mode: managed-edge
  master:
    host: <hostname>
    port: 4200
    authToken: <token>
    tls:
      disabled: false
      rejectUnauthorized: false
      requestCert: false
    resiliency: none
  group: default_fleet
```

Copy

After you've enabled TLS on the Leader, generating bootstrap scripts to add or update Edge Nodes will automatically use the `tls://` protocol for the Leader's URL.

Secure via Environment Variables

Another way to set up secure communications between Edge Nodes and the Leader is via environment variables (Stream, Edge).

If you deploy your Edge Nodes in a container, you can enable encrypted TLS communications with the Leader by configuring the `CRIBL_DIST_LEADER_URL` with the `tls:` protocol. This will override the default setting in `instance.yml`. Here's the format:

```
CRIBL_DIST_LEADER_URL=tls://<authToken>@leader:4200
```

Secure the Leader Node

This is a best practice that enables the Leader to validate itself to clients. You can secure it using a certificate you've imported:

1. Navigate to **Settings** > **Global** > **General Settings** > **API Server Settings** > **TLS**.
2. Toggle **Enabled** on.
3. From the **Certificate Name** drop-down, select a certificate you've previously imported. This will populate the corresponding fields here.
4. Click **Save**.



After this save, you must prepend `https://` to all Cribl Edge URLs on the Leader Node. For example, to get back to the Settings page you just configured, you'll now need to use `https://<hostname>:<port>/settings/system`.

Note: The `https://` protocol applies to UI/API access (port 9000). For Leader  Edge Node communication (port 4200), use the `tls://` protocol in `CRIBL_DIST_LEADER_URL`.

For detailed steps on enabling TLS for the Leader UI/API, see Configure TLS for API and UI Access.

11.3.1. How To Secure The Auth Token For The Leader Node

For new installations, the Leader Node generates its own secure, random auth token. The Leader also creates a new token when you switch from Single-instance to the Distributed mode by configuring a Node to be a Leader Node.



In versions older than 4.5.0, the default token was `criblmaster`. This default auth token does not change automatically if you upgrade your deployment to version 4.5.0 or newer.

If you need to change your auth token (for example, you prefer to use your own token or you are still using the older default of `criblmaster`), change the token value to a unique secure value.

This applies whether you configure your Leader via the UI, or via a Docker Compose file.



High Availability (Failover) Mode

Beginning in Cribl Edge 4.7, the auth token information that is stored in the `leader.yml` file on the failover volume will take precedence over the auth token stored in the local `instance.yml` file. This way, a single token is shared between primary and any standby Leaders.

In versions older than 4.7, there is no `leader.yml` file; instead, the auth token is in the `instance.yml` file only. Whenever you change an auth token, you must update it to be identical on all Leaders. See Configuring Additional Leader Nodes in the failover topic.

Change the Auth Token Value



Changing the auth token once you have Edge Nodes deployed will break communication between the Nodes and the Leader. If you do change the auth token value, make sure that you update the auth tokens for the Leader and the Edge Nodes to the same value.

Create a unique token value with strong security. We recommend creating a string that contains at least 14 characters and includes uppercase letters, lowercase letters, and numbers.

Allowed Characters in the Leader Auth Token

Starting in version 4.6.1, these are the only allowed characters you can use in the auth token for your Leader:

- a-z: Lowercase letters from a to z.

- A-Z: Uppercase letters from A to Z.
- 0-9: Digits from 0 to 9.
- `_`: Underscore.
- `!`: Exclamation mark.
- `-`: Hyphen (dash).

In version 4.5.1 and newer, you *can't* use these special characters in the Leader auth token:

- `<`: Open angle bracket.
- `>`: Close angle bracket.
- `"`: Double quotation mark.
- `{`: Open brace.
- `}`: Close brace.
- `|`: Pipe.
- `\`: Backslash.
- `^`: Caret.
- `'`: Single quotation mark.
- `` ` ``: Backtick.
- Whitespace

One option is to generate a token value with a shell command. For example, on Linux or Mac OS, the following command generates a 32-character value:

```
head /dev/urandom | LC_ALL=C tr -dc A-Za-z0-9 | head -c32 | cut -c 1-
```

Copy

Use the CLI, UI, or environment variables to change the auth token. Manually editing the `instance.yml` file to change the auth token, although technically possible, will leave the token in plain text and prevent it from being properly encrypted. Using the following methods ensures your token is securely encrypted.

Once you have created a secure auth token value, you can change it:

- In the **Auth token** field in the UI.
- In the appropriate place in your Docker Compose file (if applicable).

- For the Leader via command line: `./cribl mode-master -u <token>`. If your Cribl Edge deployment uses High Availability (HA), you must use the command line (not the UI) with the command listed above.
- For the Edge Nodes via command line:
  - `./cribl mode-managed-edge -u <token>` for Cribl Edge.
  - `./cribl mode-worker -u <token>` for Cribl Stream.



Running either of the commands resets all the related options. This means that, in addition to the token, you must provide the command with the current values of all the other arguments.

You can find the **Auth token** setting by navigating to **Settings** > **Global** > **System** > **Distributed Settings** > **Leader Settings**.

11.3.2. TLS Defaults And System-Wide Settings

Cribl Edge prioritizes secure communication by default. This ensures the integrity and confidentiality of data transmitted between your Cribl Edge instance and various Sources and Destinations.

TLS Version Support and Defaults

The minimum supported TLS version in Cribl Edge is `TLS 1.2` by default. This default applies to all secure connections, including supported Sources and Destinations, Worker/Leader communications, and Cribl.Cloud.

Cribl.Cloud and on-prem deployments use the same TLS settings, with `TLS 1.2` being the default minimum version.



You can check whether a Source or Destination supports TLS by reviewing the information box at the beginning of its corresponding documentation page.

Deployment Scope for Defaults

The TLS version and cipher defaults described on this page apply to both Cribl.Cloud and on-prem Cribl Edge deployments. However, the **Encryption: Default disabled** values in the TLS Settings and Traffic Types table reflect the base defaults for on-prem deployments only.

For Cribl.Cloud Leader deployments, TLS is enabled and managed by the Cribl.Cloud platform by default for Leader API/UI access and Leader–Worker communications. You do not need to manually enable TLS on the Leader in Cribl.Cloud. For details, see Secure Leader/Worker Nodes Communication.

TLS Settings and Traffic Types

This table shows TLS client/server pairs, and encryption defaults per traffic type. The **Default disabled** values apply only to on-prem deployments.

Traffic Type	TLS Client	TLS Server	Encryption	Cert Auth	CN* Check
UI	Browser	Cribl Edge	Default disabled	Default disabled	Default disabled
API	Worker/Edge Node	Leader	Default disabled	Default disabled	Default disabled

Traffic Type	TLS Client	TLS Server	Encryption	Cert Auth	CN* Check
Worker-to-Leader	Worker/Edge Node	Leader	Default disabled	Default disabled	Default disabled
Data	Any data sender	Cribl Edge (Source)	Default disabled	Default disabled	Default disabled
Data	Cribl Edge (Destination)	Any data receiver	Default disabled	Default disabled	Default disabled
Authentication	——–	——–	——–	——–	——–
Local*	Browser	Cribl Edge	Default Disabled	N/A	N/A
LDAP*	Cribl Edge	LDAP Provider	Custom	N/A	Default Disabled
Splunk*	Cribl Edge	Splunk Search Head	Default Enabled	N/A	Default Disabled
OIDC†/Okta*	Browser and Cribl Edge	Okta	Default Enabled	N/A	Enabled (Browser)
OIDC†/Google*	Browser and Cribl Edge	Google	Default Enabled	N/A	Enabled (Browser)

** Common name*
† OpenID Connect

System-wide TLS Settings Including Ciphers

You can configure advanced, system-wide TLS settings - minimum and maximum TLS versions, default cipher lists, and ECDH curve names. Select **Settings** > **Global** > **System > General Settings > Default TLS Settings**.

In the **Default cipher list** field, you can specify one or more ciphers from the following list:

- `ECDHE-RSA-AES128-GCM-SHA256`
- `ECDHE-ECDSA-AES128-GCM-SHA256`
- `ECDHE-RSA-AES256-GCM-SHA384`
- `ECDHE-ECDSA-AES256-GCM-SHA384`
- `DHE-RSA-AES128-GCM-SHA256`
- `ECDHE-RSA-AES128-SHA256`
- `DHE-RSA-AES128-SHA256`

- `ECDHE-RSA-AES256-SHA384`
- `DHE-RSA-AES256-SHA384`
- `ECDHE-RSA-AES256-SHA256`
- `DHE-RSA-AES256-SHA256`
- `HIGH`
- `!aNULL`
- `!eNULL`
- `!EXPORT`
- `!DES`
- `!RC4`
- `!MD5`
- `!PSK`
- `!SRP`
- `!CAMELLIA`

CA Certificates and Environment Variables

For any Cribl Edge Source or Destination that supports TLS, you can configure a **CA Certificate Path** field that points to a Certificate Authority (CA) `.pem` file(s). However, you can also use environment variables to manage CAs globally. Here are some common scenarios:

- **Add a set of trusted root CAs to the list of trusted CAs that Cribl Edge trusts.** Set the `NODE_EXTRA_CA_CERTS` environment variable for each Edge Node. For example, if you are using systemd, add the following line in each Edge Node's systemd unit file (replace `/<path>/<to>/<the>/<directory>/<containing>/<certs>/ca.pem` with the path to your CA `.pem` file):

```
...
[Service]
Environment="NODE_EXTRA_CA_CERTS=/<path>/<to>/<the>/<directory>/<containing>/<ce
...
```

Copy

When configuring TLS authentication on Edge Nodes, make sure you place your certificates into a separate directory outside of $CRIBL_HOME. If you place the certificates inside $CRIBL_HOME,

they'll be removed when the next config bundle is deployed from the Leader.

For details about `NODE_EXTRA_CA_CERTS`, see the node.js documentation.

- **Configure Cribl Edge to accept all TLS certificates, regardless of their validity**. Set the `NODE_TLS_REJECT_UNAUTHORIZED` environment variable for each Edge Node. For example, if you are using systemd, add the following line in each Edge Node's systemd unit file:

```
...
[Service]
Environment="NODE_TLS_REJECT_UNAUTHORIZED=0"
...
```

Copy

`NODE_TLS_REJECT_UNAUTHORIZED=0` disables TLS certificate validation, which can decrease the security posture of your Cribl installation. For this reason, we recommend avoiding `NODE_TLS_REJECT_UNAUTHORIZED=0` in production environments. Instead, use the `NODE_EXTRA_CA_CERTS` environment variable to explicity trust the necessary certificates.

For details about `NODE_TLS_REJECT_UNAUTHORIZED`, see the node.js documentation.

As a result of a Known Issue, passphrases in TLS certificates configured for Sources or Destinations in a Cribl Edge Fleet will not be decrypted correctly when a Subfleet inherits the configuration.

Avoid using passphrases with TLS certificates if you intend to have Subfleets inherit Source or Destination configurations.

Global CA Certificates and Sources

While Cribl Edge supports using environment variables like `NODE_EXTRA_CA_CERTS` to add trusted root CAs globally, this approach is not applicable to all Sources. Some Sources, such as the Windows Events Fowarder (WEF), require a specific CA certificate configured at the Source level to function correctly.

Additionally, OS certificate stores accessed through Node.js boot flags (such as `--use-bundled-ca`, `--use-openssl-ca`) are not supported by Cribl Edge. These flags are stripped during application startup.

In summary, certain Sources, like the WEF, exclusively rely on the specific CA certificate configured within the Source itself.

11.3.3. Secure Cribl.Cloud With TLS And Mutual TLS

Secure data transmission is essential for protecting sensitive information as it travels across networks. Cribl.Cloud offers two primary methods for securing data in transit:

- **Transport Layer Security (TLS)** - Encrypts data during transmission and authenticates the server.
- **Mutual TLS (mTLS)** - Adds client authentication on top of standard TLS.

For encrypting data in transit and simple server verification, Standard TLS is sufficient. Opt for mTLS when security policies mandate additional client verification.

This guide explains both security methods and walks you through configuration options in Cribl.Cloud.

TLS in Cribl.Cloud

In Cribl.Cloud, you can enable TLS encryption and use Cribl-provided certificates for certain Sources to ensure that your data is protected in transit.

Cribl-provided certificates offer the following benefits:

- **Reliable, ready-to-use security**: When you enable TLS and enter the correct environment variables for Cribl-provided certificates in the Source configuration, data in transit is always encrypted.
- **Zero maintenance**: Cribl-provided certificates are automatically managed and renewed.
- **Simplified deployment**: No need to generate or install custom certificates.



You only need to use your own private key or configure mTLS if specifically required by your security team or company policies. If you need custom certificates, see Certificate-Based Authentication.

While TLS encryption ends at individual Nodes upon data arrival in Cribl.Cloud, data at rest is protected by other robust security measures within the platform.

Cribl-provided certificates are unique to each Workspace and Organization.

Cribl.Cloud supports Cribl-provided certificates for Sources including Cribl TCP, Cribl HTTP, Syslog, HTTP, and Splunk TCP. To view a complete list of the supported Sources, along with their corresponding ports and ingress addresses:

1. Log in to your Cribl.Cloud portal.
2. On the top bar, select **Products**, and then select **Workspace** under **Cribl**.
3. In the sidebar, select **Data Sources**.

Enable TLS on Sources Using Cribl-Provided Certificates

To enable TLS using Cribl-provided certificates for a supported Source, configure the following **TLS Settings** for the Source:

- **Enabled**: Toggle on.
- **Private key path**: Enter `$CRIBL_CLOUD_KEY`. This environment variable represents the path to the default private key in Cribl.Cloud.
- **Certificate path**: Enter `$CRIBL_CLOUD_CRT`. This environment variable represents the path to the default certificate in Cribl.Cloud.
- **Minimum TLS version**: Select `TLSv1.2`.

When you finish configuring the Source, select **Save**, then **Commit** and **Deploy**.

For details on how to securely forward syslog data to your Cribl.Cloud instance, see Syslog TLS to Cribl.Cloud (Palo Alto Example).This topic demonstrates a vendor-specific configuration (Palo Alto) for secure data ingestion.

Enable mTLS Authentication on Cribl.Cloud

When standard TLS isn't sufficient and you need client authentication, you can enable mTLS.

Prerequisites for mTLS

mTLS requires:

- A CA certificate chain that can validate the client certificates.
- Client certificates issued to each client that will connect to Cribl.Cloud.
- Client systems configured to present their certificates during connection.

In Cribl.Cloud, you configure mTLS authentication separately for each Source.

This requires a CA certificate chain that can validate the client certificate used for authentication. For secure communication using mTLS authentication in Cribl.Cloud, you need to provide the root CA certificate of your client certificate. If you don't have the root CA certificate, you can also provide a CA certificate chain that includes all intermediate certificates leading back to a trusted root certificate.

Configure mTLS on a Source

You can add your CA certificate by creating a new certificate entry in Cribl.Cloud.

1. Prepare the CA certificate chain PEM file.
2. Go to the Edge Node: **Settings** > **Global** > **Security** > **Certificates** and select **Add Certificate**.
3. Populate the **Certificate** field with a valid PEM-formatted content (from your trusted CA certificate chain):

```
-----BEGIN CERTIFICATE-----
CERTIFICATE CONTENT
-----END CERTIFICATE-----
```

Copy

4. Populate the **Private key** with the key in PEM format:

```
-----BEGIN RSA PRIVATE KEY-----
HIDDEN PRIVATE KEY
-----END RSA PRIVATE KEY-----
```

Copy

5. In the **CA certificate** field, enter your PEM-formatted certificate and save. This will generate the CA certificate path.



While the **Add Certificate** form requires you to provide a Private Key and Certificate, these fields are not necessary for client certificate validation. This is a design limitation that requires you to provide some properly formatted values to proceed. You can use a self-signed certificate for this purpose.

Apply the CA Certificate to Your Source:

1. Edit the certificate again to view the certificate path and copy it.
2. Navigate to the Source where you want to enable mTLS and open the configuration modal. In the **TLS Settings** tab:
    1. Toggle **Enable TLS** to `Yes`.
    2. Set **Authenticate Client** to `Yes`.
    3. Paste the path in the **CA certificate path** field.
    4. Select **Save**, then **Commit** & **Deploy**.

11.3.4. Configure MTLS Authentication On-Prem

Once you have configured the Leader for encrypted TLS communication, you can implement client certificate exchange to enable mutual authentication (mTLS). This allows Cribl Edge to permit only explicitly authorized clients, which hold valid certificates, to connect to the Leader.

When a client certificate is presented to a Leader, two things happen:

1. The Leader validates the client certificate presented to Cribl Edge. The `Validate Client Certs` setting is optional, but highly recommended. When enabled, the Leader checks the certificate against the trust store, to see if it has been signed by a valid certificate authority (CA).

The Leader checks the list of certificates in the `CA Certificates Path` box first (if populated), then against the list of built-in system certificates.

2. The Leader checks whether the `Common Name` (CN) matches the regular expression in the configuration. Cribl Edge's default is to accept any value in the `Common Name` field. You can customize this as needed.

Within the `Common Name`, we validate against the value after the `CN=string`. If your `Common Name` is `CN=logstream.worker`, you would enter `logstream\.worker` in the `Common Name` field - including the backslash, because the value entered is a regular expression.



The Cribl.Cloud Leader connections on port `4200/tcp` do not currently support mutual TLS (mTLS).

Certificate Requirements for mTLS

mTLS provides a significantly higher security posture by requiring both the server and the client to prove their identities. For this handshake to succeed, Cribl Edge relies on the underlying TLS stack to validate the integrity and purpose of each certificate.

Before enabling mTLS, verify that your certificate infrastructure meets the standards outlined below. If a certificate is expired, uses a legacy algorithm, or cannot be traced back to a trusted root, the TLS connection will be rejected immediately to prevent unauthorized access.

General Certificate Standards

Regardless of their role in the handshake, all certificates used by the Leader, Worker/Edge Nodes, and Sources/Destinations must adhere to these baseline criteria:

- **Validity Period**: Certificates must be current. TLS handshakes will fail if the system time is outside the NotBefore or NotAfter window.
- **Modern Cryptography**: Use certificates signed with modern algorithms (such as `RSA` or `ECDSA`). Avoid legacy or deprecated hash functions like `MD4`, `MD5`, or `SHA-1`, which are often blocked by TLS.
- **Chain of Trust**: The peer validating the certificate must be able to build a complete path from the end-entity certificate through any Intermediate CAs to a trusted root. In Cribl Edge, this trust is established via the **CA Certificates Path** configuration or the host system trust store.

Server-Specific Requirements

Server certificates are presented by components that terminate TLS (such as the Leader on its management port). These certificates must prove they are authorized to act as a "server" to the connecting client:

- **Key usage**: The certificate Extended Key Usage (EKU) must include `serverAuth`.
- **Hostname Verification**: To prevent man-in-the-middle attacks, the Common Name (CN) or Subject Alternative Name (SAN) must exactly match the `hostname` or IP address that clients use to connect (for example, `leader.cribl.com`).

Client-Specific Requirements

Client certificates are presented by the component initiating the connection (such as a Worker/Edge Node connecting to a Leader). These are subject to stricter identity checks:

- **Client Authentication**: The certificate EKU must explicitly include `clientAuth`. Public CAs (such as Let's Encrypt) have increasingly disallowed this attribute in their standard issuance policies to comply with stricter industry root program requirements. For mTLS, you will likely need to use an internal PKI or a specialized certificate product that permits client-side usage.
- **Identity Constraints**: Client certificates must be leaf certificates (where **Basic Constraints** are set to `CA=false`). Root or Intermediate CA certificates should never be used as client identifiers.
- **CN Regex Matching**: Cribl Edge provides an additional layer of security by checking the certificate CN against a user-defined regular expression to ensure only specific named clients are permitted.

Limitations on mTLS Authentication

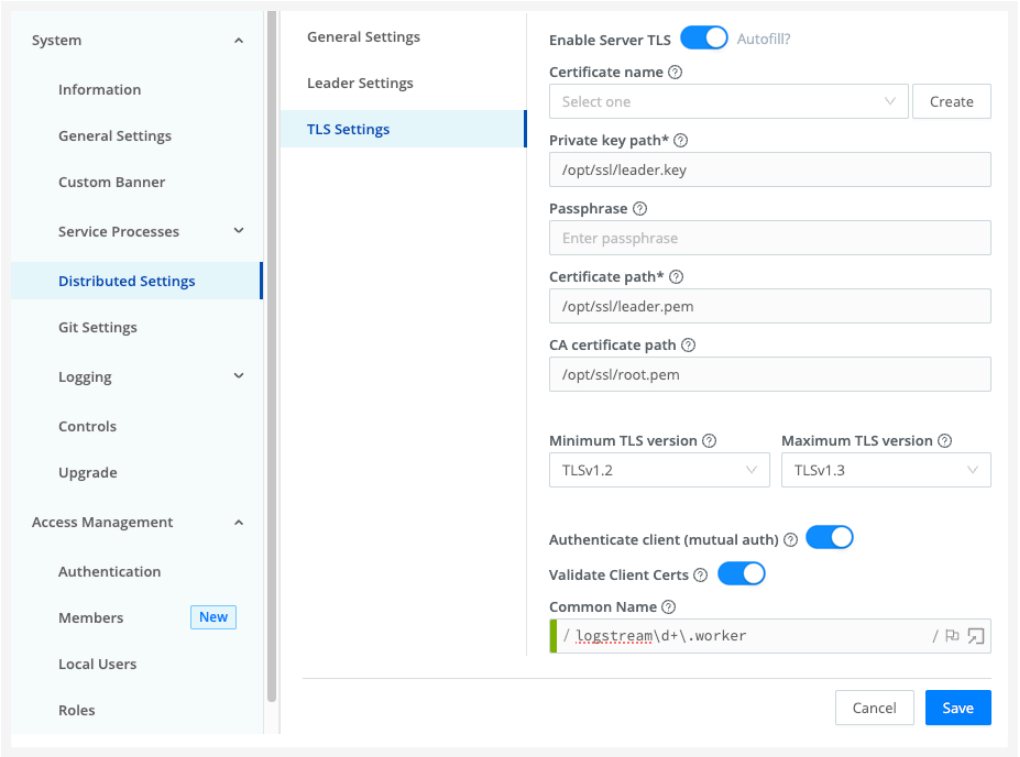
When configuring mTLS authentication on Edge Nodes, make sure you place your certificates into a separate directory outside of `$CRIBL_HOME`. If you place the certificates inside `$CRIBL_HOME`, they'll be removed when the next config bundle is deployed from the Leader.

Similarly, you can't bootstrap Edge Nodes with mutual authentication already populated. To bootstrap Edge Nodes, you supply only the shared authentication token. Certificates should be viewed as two-factor authentication; so placing the certificates in the config bundle defeats the purpose of two-factor authentication.

Configure via the UI

mTLS authentication in Cribl.Cloud is configured separately for each Source.

Below is a sample configuration on the Leader:



Configuring mutual authentication

In the above configuration, note:

- You can set both the **Minimum TLS version** and **Maximum TLS version**.
- The `Common Name` regex contains an additional check for `\d+`. This allows checking for the format: `logstream1.worker`, `logstream2.worker`, `logstream3.worker`, etc.

Using YAML Config File

To set up mTLS authentication via the Edge Node's `instance.yml` config file, you need to change some values and also add a few keys, as shown in this example:

```
distributed:
  mode: managed-edge
  master:
    host: <hostname>
    port: 4200
    authToken: <token>
    tls:
      disabled: false
      rejectUnauthorized: true # false if ignoring untrusted certs
      requestCert: true
      privKeyPath: /path/to/certs/worker.key
      certPath: /path/to/certs/worker.pem
      caPath: /path/to/certs/root.pem
    resiliency: none
  group: default_fleet
```

Copy

Using Environment Variables

You can set up mTLS authentication by configuring this environment variable, using the format shown in this example:

```
CRIBL_DIST_LEADER_URL="tls://<authToken>@leader.cribl:4200?tls.privKeyPath=/path/to/c
```

Copy

Once you've set this variable, restart the Edge Node. You should see the Edge Node successfully reconnect to the Leader.

If the Edge Node doesn't connect, check `cribl.log` on both the Edge Node and Leader for more context about the problem. You should see errors related to `dist leader communications`.



To build your own Certificate Authority (a self-signed CA), see our blog post on How to Secure Cribl Stream Worker-to-Leader Communications.

11.3.5. Import Certificates And Keys

This topic explains how to import and manage TLS/SSL certificates and keys in Cribl Edge. Certificates you import here can be reused across multiple purposes, including:

- Secure TLS for API and UI Access
- Securing Leader–Edge Node Communication
- Secure Sources and Destinations with Certificates

Certificate Requirements

Cribl Edge expects certificates and keys to be formatted in privacy-enhanced mail (`.pem`) format.

You can use:

- Certificates from a public certificate authority (recommended for production; trusted by most clients by default).
- Certificates issued by your organization internal PKI (requires adding the root and intermediate CA certificates to client trust stores).
- Self-signed certificates (for lab/test environments only; requires manual trust configuration).

Generate a Self-Signed Certificate (Optional)

For non-production or lab environments, you can generate a self-signed certificate and private key using OpenSSL:

```
openssl req -nodes -new -x509 -newkey rsa:2048 \
  -keyout myKey.pem \
  -out myCert.pem \
  -days 420
```

Copy

This command generates:

- `myCert.pem`: a self-signed certificate (valid for 420 days).
- `myKey.pem`: an unencrypted 2048-bit RSA private key.

Import a Certificate

On the Leader

1. Navigate to **Settings** > **Global** > **Security** > **Certificates**.
2. Select **Add Certificate**.
3. Enter a descriptive **Name** (for example, `leader-ui-tls` or `worker-tls`).
4. In the **Certificate** field, do one of the following:
    - Drag and drop your `.pem` certificate file into the field.
    - Click the upload button to browse for the file.
    - Open the certificate file, copy its contents, and paste them into the field.
5. In the **Private key** field, upload or paste the corresponding private key in PEM format.
6. (Optional) If your private key is encrypted, enter the **Passphrase**.
7. (Optional) If you're using a certificate signed by an external certificate authority (for example, a downloaded Splunk Cloud certificate), upload or paste the CA certificate chain into the **CA certificate** field. For details, see Obtain the Certificate Chain (TLS/SSL).
8. (Optional) Add a **Description** to help identify the certificate's purpose.
9. Select **Save**.

On a Fleet

For distributed deployments, you can also import certificates at the Fleet level:

1. In the sidebar, select **Fleets**, then select the Fleet you want to configure.
2. Select **Fleet Settings**.
3. Navigate to **Security** > **Certificates**.
4. Select **Add Certificate**.
5. Follow the same steps as above to upload your certificate and private key.
6. Select **Save**.
7. **Commit and Deploy** the Fleet's new configuration.

Manage Existing Certificates

To view and manage certificates you've already imported:

1. Navigate to **Settings** > **Global** > **Security** > **Certificates** (Leader) or **Fleet Settings** > **Security** > **Certificates** (Fleet level).
2. Select a certificate name to view or edit its details.
3. To delete a certificate, click the delete icon next to the certificate entry.



Before deleting a certificate, verify it's not in use by any TLS configuration (Leader UI/API, Leader–Edge Node communications, Sources, or Destinations).

Using Imported Certificates

After importing a certificate, you can reference it by name in various TLS configurations:

Purpose	Where to Configure
Leader UI/API	**Settings** > **Global** > **General Settings** > **API Server Settings** > **TLS** > **Certificate name**
Leader–Edge Node Communications	**Settings** > **Global** > **System** > **Distributed Settings** > **TLS Settings** > **Certificate name**
Edge Node TLS	**Edge Node Settings** > **System** > **Distributed Settings** > **TLS Settings** > **Certificate name**
Sources/Destinations	Individual Source or Destination TLS settings

For step-by-step instructions on enabling TLS for specific use cases, see:

- Configure TLS for API and UI Access
- Secure Leader/Edge Nodes Communication

11.3.6. Configure TLS For API And UI Access

To safeguard inbound communications to your Leader UI and API (default port 9000), enable Transport Layer Security (TLS) on the Leader.

> This topic applies to on-prem and customer-managed Leader deployments only. For Cribl.Cloud deployments, TLS is enabled by default and managed automatically by the Cribl.Cloud platform.

This topic focuses on securing the Leader UI/API listener. For other TLS configurations, see:

- Secure Leader/Edge Nodes Communication: Secure the Leader–Edge Node control-plane channel (default port 4200).
- Import Certificates and Keys: Import and manage TLS certificates for reuse across multiple purposes.

Prerequisites

Before you enable TLS for the Leader UI/API, you need a TLS/SSL certificate and private key imported into Cribl Edge. For details on generating or importing certificates, see Import Certificates and Keys.

Step 1: Enable TLS for the Leader UI/API

After you have a certificate imported at **Settings** > **Global** > **Security** > **Certificates**, enable TLS for the Leader UI/API listener:

1. In the Leader UI, go to **Settings** > **Global** > **General Settings** > **API Server Settings** > **TLS**.
2. Toggle **Enabled** on.
3. In **Certificate name**, select a certificate you've imported (for example, `leader-ui-tls`).
4. (Optional) Adjust any additional TLS options as needed (for example, minimum/maximum TLS versions).
5. Select **Save**.

After you save:

- The Leader UI/API now listens on HTTPS.
- Access the UI/API using: `https://<hostname>:<port>/`
  For example: `https://leader.example.com:9000/settings/system`

After enabling TLS, you must prepend `https://` to all Cribl Edge URLs on the Leader Node.

Step 2 (Optional): Enable TLS for Leader–Edge Node Communications

To secure the Leader–Edge Node control-plane channel (port 4200), see Secure Leader/Edge Nodes Communication. That guide walks you through:

- Uploading the same certificate (or a different one) to the relevant Fleets.
- Enabling TLS on each Edge Node's distributed settings.
- Enabling TLS for the Leader's distributed endpoint so Edge Nodes connect securely.

Advanced: Configure TLS via Config Files

Most deployments should configure TLS through the UI. Use this method only when you have a clear config-as-code workflow. Avoid ad-hoc manual edits of local config files on running systems; prefer the UI or a consistent automation pipeline that owns these files.

If you manage Cribl Edge via automation (for example, configuration management, containers, or Cribl as Code), you can configure the Leader UI/API listener in `cribl.yml`.

Example `cribl.yml` snippet:

```
api:
  host: 0.0.0.0
  port: 9000
  disabled: false
  ssl:
    disabled: false
    privKeyPath: /path/to/myKey.pem
    certPath: /path/to/myCert.pem
```

Copy

For details on generating and importing certificates and keys used in this example, see Import Certificates and Keys.

Restore Unencrypted Private Keys

In Cribl Edge 4.1 and newer, for enhanced security, Cribl encrypts TLS certificate private keys in configuration files when you add or modify them. Before upgrading to v.4.1.0 (or newer), back up your existing unencrypted key files – see Safeguarding Unencrypted Private Keys for Rollback.

If you need to roll back from Cribl Edge 4.1 or later to an earlier version, follow this procedure to restore the unencrypted private key files that you backed up earlier:

1. Stop Cribl Edge 4.1 (or later) on hosts – the Leader and all Edge Nodes.
2. Untar the older Cribl Edge version (v.4.0.4, for example) onto the Leader and all Edge Nodes.
3. Manually edit each Cribl Edge config file that contains an encrypted private key. Replace each key with its prior unencrypted version.
4. Start up Cribl Edge, commit and deploy, and resume normal operations.

11.3.7. Secure Sources And Destinations With Certificates

You can secure connections between Cribl Edge and external systems by using TLS (Transport Layer Security) certificates or custom HTTP headers. These extra security measures are optional, but TLS certificates or custom HTTP headers help authenticate data senders and receivers, protect data in transit, and prevent unauthorized access.

Certificate-Based Authentication

Cribl Edge stores and manages certificates at the Fleet level. When you add a certificate to a Source or Destination, Cribl Edge creates it within the same Fleet where you configured the integration.

For information about how certificate references work in Packs, see Use Named TLS Certificates.

Add a TLS Certificate to a Fleet

To add a TLS certificate that will be available to all Sources and Destinations at the Fleet level:

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet.
2. Navigate to **Fleet Settings**. On the side menu, expand **Security** and select **Certificates**.
3. Select **Add Certificate**. In the **New Certificate** modal:
    - **Name:** Enter a unique name to identify the certificate.
    - **Description:** Optionally, add a short description of the certificate's scope or purpose.
    - **Certificate:** Upload or paste the full contents of your certificate file. Ensure you paste the entire block, including the `BEGIN` and `END` lines:

      ```
      -----BEGIN CERTIFICATE-----
      ... (certificate data) ...
      -----END CERTIFICATE-----
      ```

      Copy

    - **Private key:** Upload or paste the full contents of your private key (`.pem`). Ensure you paste the entire block, including the `BEGIN` and `END` lines. Some private keys may use labels like `RSA PRIVATE KEY` or `EC PRIVATE KEY`. Be sure to include the correct `BEGIN` and `END` lines that match your key type.

- **Passphrase:** Optionally include the certificate's passphrase if it uses passphrase encryption.
- **CA certificate:** Optionally upload or paste a certificate signed by an external certificate authority by importing the chain. See Obtain the Certificate Chain (TLS/SSL) for more information.
- **Referenced:** This table shows which Sources and Destinations are using this certificate. If this table is empty, you haven't assigned this certificate to any components yet.

4. Select **Save**. The certificate now appears in the list of certificates for the Fleet.

Add a TLS Certificate to a Source or Destination

To add a TLS certificate directly to an integration:

1. Open the configuration settings for your Source or Destination.
2. Select **TLS Settings (Client Side)** or **TLS Settings (Server Side)**, depending on the integration.
3. Toggle **Enabled** on.
4. Use the **Certificate name** drop-down to add a certificate:
   - **To add a new certificate:** Select the **Create** button next to the field. See Add a TLS Certificate to a Fleet for field descriptions.
   - **To add an existing certificate:** Select the name of the certificate from the drop-down. This will auto-populate the related fields.
5. Select **Save**.

Maintain TLS Certificates

Certificate revocation can lead to unexpected service interruptions. To mitigate risks:

- Regularly review and renew certificates.
- Implement robust monitoring and alerting systems to detect potential issues.
- Ensure secure storage of certificates and private keys.
- In case of emergency, have a well-defined recovery plan to restore service quickly.

Custom HTTP Headers

You can also encode custom, security-related HTTP headers as needed. In a Distributed deployment, navigate to Worker Group Settings, then **API Server Settings** > **Advanced** > **HTTP Headers**. (In a Single-instance deployment, the **API Server Settings** are within **Global Settings** > **General Settings**.) Define HTTP headers as key-value pairs, selecting **Add Header** as needed.

Implement a Custom Content Security Policy (CSP)

Content Security Policy (CSP) helps protect Cribl Edge from Cross-Site Scripting (XSS) and other web injection attacks by controlling which resources the browser is allowed to load. You can implement CSP using the Custom HTTP Headers setting described above.

Basic Configuration

To configure a basic CSP for Cribl Stream, add the following header in your **HTTP Headers** settings:

Header Name: `Content-Security-Policy`

Header Value:

```
default-src 'self'; script-src 'self' 'unsafe-eval' 'sha256-ZaAN8+jAf3MdnSLRdmTuExUhA
```

Copy

Test your CSP policy thoroughly in a development environment before deploying to production and monitor the browser console for CSP violations after implementation.

SAML SSO Configuration

If your organization uses SAML Single Sign-On, you'll need a modified CSP policy to accommodate the dynamic JavaScript and form submissions required for SAML authentication:

Header Name: `Content-Security-Policy`

Header Value for SAML:

```
default-src 'self'; script-src 'self' 'unsafe-eval' 'unsafe-inline'; connect-src 'sel
```

Copy

Replace `YOUR_IDP_DOMAIN` with your actual Identity Provider's domain (for example, your Okta domain).

SAML configurations require additional permissions that reduce the restrictiveness of the CSP compared to the basic configuration.

11.3.8. Configure System Proxy

Configure the proxy server to handle HTTP/S requests.

You can direct all outbound HTTP/S requests to go through proxy servers. You do so by setting environment variables before starting Cribl Edge.

Learn how to configure environment variables.

See System Proxy Configuration on Windows for details of Windows-specific proxy configuration.

Configure the `HTTP_PROXY` and `HTTPS_PROXY` environment variables, either with your proxy's IP address, or with a DNS name that resolves to that IP address. Optionally, follow either convention with a colon and the port number to which you want to send queries. Ensure you explicitly include the protocol (`http://` or `https://`) in the variable values.

`HTTP_PROXY` examples:

```
HTTP_PROXY=http://10.15.20.25:1234
HTTP_PROXY=http://proxy.example.com:1234
```

Copy

`HTTPS_PROXY` examples:

```
HTTPS_PROXY=http://10.15.20.25:5678
HTTPS_PROXY=http://proxy.example.com:5678
```

Copy

In the above examples, note that when you set an `HTTPS_PROXY` environment variable, the referenced URL should generally be in `http` format.

To set up Leader/Worker communication, use the CRIBL_DIST_WORKER_PROXY environment variable.

Restarts and Case Conflicts

> Initial configuration of, and changes to, these variables require restarting Cribl Edge on the affected Nodes, if the application is already running when you apply the changes.
>
> The environment variables' names can be either uppercase or lowercase. However, if you set duplicate versions of the same name, the lowercase version takes precedence. For example, if you've set both `HTTPS_PROXY` and `https_proxy`, the IP address specified in `https_proxy` will take effect.

HTTP and/or HTTPS?

Several Cribl Edge endpoints rely on the HTTPS protocol - the Cribl telemetry endpoint, which must be accessed with some license types, as well as the CDN used to propagate application updates and certain documentation features (API Reference and docs PDFs).

You might configure certain other Cribl Edge features (such as REST API Collectors) that require access to HTTP endpoints. For maximum flexibility, consider setting environment variables to handle both the HTTPS and HTTP protocols.

Authenticating on Proxies

You can use HTTP Basic authentication on HTTP or HTTPS proxies. Specify the user name and password in the proxy URL. For example:

```
HTTP_PROXY=http://username:password@proxy.example.com:1234
HTTPS_PROXY=http://username:password@proxy.example.com:5678
```

Copy

Proxies and Certificates

If a proxy presents its own certificate, Cribl Edge must trust the entire certificate chain.

To ensure the entire chain is available:

1. Gather all the required CA certificates in PEM format, including the root and intermediate certificates.
2. Create a single PEM file containing all the certificates, in the correct order, starting with intermediate certificates, ending with the root. You can do it with the follow command:

```
$ cat intermediate1.pem intermediate2.pem root_ca.pem > my_proxy_ca_chain.pem
```

Copy

3. Place the PEM file in a directory outside `$CRIBL_HOME`.

Bypassing Proxies with `NO_PROXY`

If you've set the above environment variables, you can negate them for specified (or all) hosts. Set the `NO_PROXY` environment variable to identify URLs that should bypass the proxy server, to instead be sent as direct requests. Use the following format:

```
NO_PROXY="<list of hosts/domains>"
```

Cribl recommends including the Leader Node's host name in the `NO_PROXY` list.

Client Configuration and Leader Reference

The way a client references the Leader Node depends on its configuration, which originates from the Distributed Settings for that particular Node. These settings reside in the `$CRIBL_HOME/local/_system/instance.yml` file.

When deploying a configuration update, the Leader Node informs all connected clients (Workers and Edge Nodes) within its Group or Fleet that a new configuration bundle is available. The Leader will reference itself using the same method (IP address or FQDN) that clients already use to connect to it in the distributed environment. This ensures a seamless download process for the clients, regardless of their initial connection method.

If a Worker is configured to use the Leader's IP for communication, the URL will also contain that specific IP. Consequently, the `NO_PROXY` variable on the client must match the IP address used in the client configuration and Leader URL for optimal operation.



Cribl v.4.6.0 introduces direct S3 downloads for configuration bundles, bypassing the Leader Node and scaling deployments with many Edge Nodes. However, if your Edge Nodes are firewalled or requires access controls, a proxy might be needed. In such scenarios, you need to exclude the S3 bucket address from the "NO_PROXY" list for successful downloads.

`NO_PROXY` Usage

Within the `NO_PROXY` list, separate the host/domain names with commas or spaces.

Optionally, you can follow each host/domain entry with a port. If not specified, the protocol's default port is assumed.

To match subdomains, you must either list them all in full (for example, `NO_PROXY="foo.example.com","bar.example.com"`), or apply a wildcard by prefixing the domain name with a period or `*.`: `NO_PROXY=".example.com"` or `NO_PROXY="*.example.com"`.

To match the whole domain including its subdomains, add it both with and without wildcard to the list: `NO_PROXY="example.com,.example.com"`.

To disable all proxies, use the `*` wildcard: `NO_PROXY="*"`. `NO_PROXY` with an empty list disables no proxies.

Cloud `NO_PROXY` Usage

You must include any cloud metadata endpoints (such as the AWS Instance Metadata Service) in the `NO_PROXY` list:

- AWS EC2 and Azure VM instances must include `169.254.169.254` in the list. If using IPv6 on AWS EC2, add `fd00:ec2::254` to the list.
- AWS ECS Fargate tasks must include `169.254.170.2`.
- GCP (Google Cloud Platform) VM instances must include `metadata.google.internal` and `169.254.169.254`.

Where Proxies Apply

Proxy configuration in Cribl Edge applies to components that initiate outbound HTTP or HTTPS connections from the Edge Node. This typically includes:

- HTTP/S Destinations that send data to remote services. For example, cloud logging platforms, object stores, and observability APIs.
- Notification targets that call external webhooks or API endpoints over HTTP/S.

Proxies are not used for data collection, such as system metrics, file monitoring, Syslog, or other Sources that do not make outbound HTTP/S requests.

Support for proxies varies by integration. For any specific Edge Destination or Notification target, see its individual documentation page to confirm whether it supports proxies.

Testing Proxies

To initially test your proxy configuration, consider setting up a simple, free proxy server like mitmproxy (https://mitmproxy.org), and then monitoring traffic through that server. Verify that you can trace proxied requests from your Cribl Edge instance, and can validate that outgoing requests (to Destinations) are working properly.

Proxying Multiple Edge Instances in One Browser

Cribl Edge stores authentication tokens based on each http header's URI scheme, host, and port information. Within a given browser, Cribl Edge enforces a same-origin policy to isolate instances.

This means that if you want to run multiple proxied Cribl Edge instances in one browser session, you must assign them different URI schemes, hosts, and/or ports. Otherwise, logging into an extra Cribl Edge instance will expire the prior instance's session and log it out.

For example, assume that you've set up this pair of Apache proxy forward rules:

- https://web/cribla forwards to `cribl_hosta:8001/cribla`.
- https://web/criblb forwards to `cribl_hostb:8001/criblb`.

These two proxied addresses cannot be run simultaneously in the same browser session. However, this pair - which lead with separate URI schemes - could:

- https://web/cribla forwards to `cribl_hosta:8001/cribla`.
- https://web2/criblb forwards to `cribl_hostb:8001/criblb`.

Where separate instances **must** share URI formats, a workaround is to open the second instance in an incognito/private browsing window, or in a completely different browser.

11.3.9. Configure SOCKS Proxy

This document outlines how to configure Cribl Edge to use a SOCKS proxy server for outbound communication.

By using a SOCKS proxy, you can enhance security, bypass network restrictions, and route specific traffic through a controlled intermediary server.



The SOCKS proxy is exclusively for Edge Nodes to communicate with a Leader Node. The Leader Node doesn't use the proxy for its outbound communications.

Architectural Overview: SOCKS Proxy in Cribl Deployments

A SOCKS proxy acts as an intermediary for outbound network connections from a client application (in this case, your Edge Node). It allows you to centralize and control the egress points for your Cribl infrastructure. The specific architectural flow depends on whether your Cribl deployment is entirely on-prem or on Cribl.Cloud.

Management vs. Data Plane Communications

If you are using a proxy for Cribl software cluster communications, it must be a SOCKS proxy (Layer 4), not an HTTP proxy. This is because cluster communications use raw TCP connectivity on port 4200, which HTTP proxies cannot handle.

Communication Types Overview

Cribl products require distinct proxy types for different communication streams. Management communications between Leaders and Edge Nodes use a proprietary protocol over persistent TCP connections (port 4200), necessitating a **SOCKS (Layer 4) proxy** for relaying.

Configuration bundle downloads, however, use standard HTTP/HTTPS protocols. If you're using a proxy for these downloads, it needs to be an **HTTP proxy** (configured via `HTTP_PROXY`/`HTTPS_PROXY` environment variables) with access to the Leader Node's (ports 443/4200) or S3 endpoints. This differs from the SOCKS proxy needed for management communications. For details, see Manage Config Bundles.

Given these different proxy requirements, you'll often find your deployment needs both SOCKS and HTTP proxies. It really depends on your network environment and whether you choose to use proxies for specific communication types.

The table below summarizes the communication type and proxy support for each.

Communication Type	Protocol Requirements	Proxy Support	Port	Purpose
Management Plane	TCP (Layer 4)	SOCKS only HTTP proxies NOT supported	`4200`	Leader ↔ Edge Node control, configuration, and status
Config Bundle Downloads	HTTP/HTTPS (Layer 7)	SOCKS proxies NOT supported HTTP proxies only (`HTTP_PROXY`/`HTTPS_PROXY`)	`443`/`4200`	Edge Node downloading configuration bundles from Leader or S3
Data Plane	HTTP/HTTPS (Layer 7)	HTTP proxies (`HTTP_PROXY`/`HTTPS_PROXY`)	Various	Data ingestion, forwarding to destinations (S3, Splunk, etc.)

On-prem Deployment Architecture

In an on-prem deployment, all Cribl components - the Leader Node and the Edge Node - are deployed and managed within your organization's network. If you choose to use a SOCKS proxy, it will exclusively control the outbound communication from your Edge Nodes to their Leader Node.

Cribl.Cloud/Hybrid Deployment Architecture

In a Cribl.Cloud deployment, the Leader Node is hosted and managed by Cribl within the Cribl.Cloud infrastructure. Your Edge Nodes, however, are typically deployed within your own on-prem network or private cloud. A SOCKS proxy can be used to route the outbound communication from your customer-managed Worker Nodes to the Cribl-managed Leader in the cloud.

Configure SOCKS Proxy

SOCKS proxy settings are applied directly to the specific Cribl Nodes (Leader, Worker, or Edge) that need to use the proxy for their outbound communication.

There are two ways to define proxy variables in Cribl Edge.

- **Environment Variables**: Ideal for most scenarios, especially when automating deployments or managing a large number of Edge Nodes.
- **UI Configuration**: Suitable for smaller deployments or when manual configuration is preferred. However, be aware of the additional steps and potential complexities involved.

Configure via Environment Variables

This is the most straightforward and flexible method, ideal for automated deployments, containerized environments (Docker, Kubernetes), or managing a large number of Nodes.

For Worker/Edge Nodes Connecting to a Leader

Use the `CRIBL_DIST_WORKER_PROXY` on each Edge Node to configure its connection to the Leader Node.

```
CRIBL_DIST_WORKER_PROXY="socks5://<username>:<password>@<proxy_host>:<proxy_port>"
```

Copy

The default protocol is SOCKS5, which uses the format: `socks5://<username>:<password>@<host>:<port>`. Specify the SOCKS proxy host, port, and optional authentication credentials in the variable.

- `<username>:<password>`: Optional authentication credentials for your SOCKS proxy.
- `<proxy_host>`: The hostname or IP address of your SOCKS proxy server.
- `<proxy_port>`: The port on which your SOCKS proxy server listens.

You can alternatively specify SOCKS4 as the protocol with the format: `socks4://<proxyhost>:<port>.` To authenticate on a SOCKS4 proxy with username and password, use this format: `socks4://<username>:<password>@<proxyhost>:<port>`. The `<proxyhost>` can be a `hostname`, `IPv4`, or `IPv6`.

For Leader Node Connections (On-Prem Specific)

In on-prem deployments, if your Leader Node needs to route its own outbound connections (e.g., to external services like CDNs for software updates, licensing servers, or telemetry endpoints) through a web proxy, configure this using standard HTTP proxy settings. Note that Leader Nodes do not initiate connections to Edge Nodes - those connections are always initiated by the Edge Nodes to the Leader. SOCKS proxy configuration is not supported for Leader Node outbound connections to cloud services.

Where to set environment variables:

- **Systemd Service:** For Linux deployments, add `Environment="<VARIABLE_NAME\>=..."` under the `[Service]` section in your systemd override file (such as, `/etc/systemd/system/cribl.service.d/override.conf`).
- **Docker/Kubernetes:** Set as environment variables in your Dockerfile, `docker run` command, or Kubernetes Deployment/StatefulSet manifests.

- **Directly on Host:** For manual installations, you can set it in a shell script that starts Cribl, or in a profile file (for example, `.bashrc`, `.profile`) for the user running Cribl.



After setting the environment variable, restart the Cribl service on the Node for the changes to take effect.



Config bundle downloads cannot be sent via a SOCKS proxy, but instead use an HTTP proxy that allows requests to the Leader or an S3 bucket.

Configure via the UI

Configuring a SOCKS proxy for Cribl Edge using the UI is more complex than using environment variables due to the multi-step process involved. It requires enabling direct access to the Leader, setting up the proxy on the Edge Nodes, and then configuring the SOCKS proxy settings. Additionally, using the UI might not be suitable for all deployment scenarios, especially those with large numbers of Edge Nodes or complex network configurations.

Configuring a SOCKS proxy for Cribl Edge using the UI is suitable for individual Edge Nodes or smaller deployments where direct access to each Edge Node's UI is feasible.

To configure a SOCKS proxy for Cribl Edge, follow these steps.

1. **Enable direct access**: Allow Edge Nodes to access the Leader Node directly.
    - Cribl Edge: Set up Leaders and Edge Nodes.
    - Cribl Stream: Set up Leaders and Worker Nodes.
    - Remember to enable UI Access (and teleport) to Workers or Edge Nodes as applicable.
2. **Set up Edge Nodes and SOCKS proxy**: Configure the Edge Nodes and ensure the SOCKS proxy server is running and accessible.
3. **Configure proxy details**: You can access the Edge Node via teleporting. To configure without teleporting, log in to each Edge Node directly using a local user account.
4. Select **Edge Node Settings** from the Edge Node submenu. Navigate to **Distributed Settings** > **Leader Proxy Settings** and configure the following fields:
    - **Username**: Enter the username for the SOCKS Proxy authentication.
    - **Password**: Enter the password for the SOCKS Proxy authentication.
    - **Proxy host**: Enter the proxy server host. Can be a `hostname`, `IPv4`, or `IPv6`.
    - **Proxy port**: The port on which your SOCKS proxy server listens.
    - **Protocol version**: Version of the SOCKS protocol. Defaults to `5` for `socks5`.

5. **Save the settings**: The Edge Nodes will automatically reconnect to the Leader node using the configured SOCKS proxy.
6. **Disable direct access**: Verify the SOCKS proxy configuration is verified. To ensure all traffic between Edge Nodes and Leader is routed through the SOCKS proxy, disable the direct network connections between them. This includes removing any firewall rules or network configurations that allow direct communication.

Best Practices & Troubleshooting

For successful SOCKS proxy implementation, consider the following.

- **Firewall Rules**: Set up your firewall rules properly for a seamless deployment.
    - **Edge Node to SOCKS Proxy:** Allow outbound TCP from Cribl Edge Node to your SOCKS proxy on its configured `proxy_port`.
    - **SOCKS Proxy to Leader (TCP 4200)**: Your SOCKS proxy must have outbound access to the Leader Node on TCP port 4200. For on-Prem deployments, ensure your Leader Node accepts inbound connections from the SOCKS proxy on TCP 4200. For Cribl.Cloud deployments, allow outbound access from your SOCKS proxy to the Cribl.Cloud Leader's IPs for the Network Load Balancers (found in **Workspace** > **Access** > **Leader NLB IPs**). For details, see View Workspace Detail.
- **DNS Resolution**: The SOCKS proxy server must resolve the Leader Node's hostname (on-prem Leader or `your-workspace.cribl.cloud`).
- **Proxy Placement (On-Prem)**: Strategically place the SOCKS proxy to optimize network paths and minimize latency.
- **Egress Control (Cribl.Cloud)**: The SOCKS proxy provides a vital, controlled egress point for Cribl Nodes' communication with Cribl.Cloud, aligning with corporate security policies.
- **Authentication**: Securely manage and ensure correct credentials if your SOCKS proxy requires authentication.
- **Other Proxy Types**: SOCKS proxies (`CRIBL_DIST_WORKER_PROXY`) are for communication between the Leader and Edge Node. Other outbound traffic (for example, to external Sources/Destinations) may require standard HTTP/HTTPS proxies (`HTTP_PROXY`/`HTTPS_PROXY`). For details, see Configure System Proxy.

11.4. Manage Secrets And Keys

Encryption keys and secrets allow you to secure data inside events, as well as authorization tokens, passwords, API keys and others.

Source	Description
Create and Manage Encryption Keys	Create and Manage Encryption Keys
Create and Manage Secrets	Protecting Secrets with Centralized Store
Configure KMS	Configuring internal and external Key Management Services

11.4.1. Create And Manage Encryption Keys

You can create and manage keys that Cribl Edge will use for real-time encryption of fields and patterns within events. For details on applying the keys that you define here, see Encryption Keys.



After setting up a key, you can also reference it in Cribl Search, using the `encrypt` and `decrypt` functions in your queries.

Access Keys

- In a Single-instance deployment, select **Settings** > **Security** > **Encryption Keys**.
- In a Distributed deployment with one Fleet, select **Settings** > **Security** > **Encryption Keys**.
- In a Distributed deployment with multiple Fleets, keys are managed per Fleet. Select **Manage** > **Groups** > `<groupName>` **Group Settings** > **Security** > **Encryption Keys**.

On the resulting **Manage Encryption Keys** page, you can configure existing keys, and/or use the following options to add new keys.

Get Key Bundle

To import existing keys, click **Get Key Bundle**. You'll be prompted to supply a login and password to proceed.

Add New Key

To define a new key, click **New Key**. The resulting **New Key** modal provides the following controls:

Key ID: Cribl Edge will automatically generate this unique identifier.

Description: Optionally, enter a description summarizing the purpose of the key.

Encryption algorithm: Currently, Cribl Edge supports the `aes-256-cbc` (default) and `aes-256-gcm` algorithms.

KMS for this key: Currently, the only option supported here is `local` (the Key Management Service built into Cribl Edge).

Key class: Classes are arbitrary collections of keys that you can map to different levels of access control. For details, see Key Classes. This value defaults to `0`; you can assign more classes, as needed.

Expiration time: Optionally, assign the key an expiration date. Directly enter the date or select it from the date picker.

Use initialization vector: If enabled, Cribl Edge will seed encryption with a nonce to make the key more random and unique. Optional (and defaults to disabled) with the `aes-256-cbc` algorithm; automatically enabled (and cannot be disabled) with the `aes-256-gcm` algorithm.

Initialization vector size: Length of the initialization vector (IV), in bytes. This option is displayed only with the `aes-256-gcm` algorithm. Defaults to `12` bytes to optimize interoperability, but you can use the drop-down to set this anywhere between `12` to `16` bytes.

Modify Keys

To protect against accidental changes, a key's parameters - once saved - can be edited only through configuration files.

When you update keys by editing the `keys.json` file, you must add them back to the directories above (respectively, on a single instance or on a Distributed deployment's Leader Node).

Sync `auth/(cribl.secret|keys.json)`

> Cribl Edge performs decryption only when the data is being sent to Splunk. For details, see Decryption of Data in Splunk

To successfully decrypt data, the `decrypt` command will need access to the same keys that were used to encrypt, **in the Cribl instance where encryption happened**.

- In a Single-instance deployment, the `cribl.secret` and `keys.json` files reside in: `$CRIBL_HOME/local/cribl/auth/`.
- In a Distributed deployment, the `cribl.secret` and `keys.json` files reside on the Leader Node in: `$CRIBL_HOME/groups/<group-name>/local/cribl/auth/`.
- When using the UI, you can download these files by selecting **Get Key Bundle**.

Sync/copy these files over to their counterparts on the Search Head/decrypting side, residing in: `$SPLUNK_HOME/etc/apps/cribl/local/cribl/auth/`.

11.4.2. Create And Manage Secrets In Cribl Edge

The Cribl Edge secrets store enables you to centrally manage secrets that Cribl Edge instances use to authenticate on integrated services. Use the **Manage Secrets** page to create and update authorization tokens, username/password combinations, and API-key/secret-key combinations for reuse across the application. These defined secrets can then be programmatically accessed and used within your Cribl Edge configurations using the C.Secret Function.

To ensure they can be decoded correctly, all secrets in Cribl Edge must use `latin1` encoding.

How Secrets Are Protected

`cribl.secret` is the master encryption key used by Cribl Edge to protect sensitive values such as credentials, tokens, and other secrets stored in configuration files. It is the root of trust for securing encrypted values across the platform and is created when Cribl Edge first starts.

File Location

By default, `cribl.secret` is stored at:

```
$CRIBL_HOME/local/cribl/auth/cribl.secret
```

Copy

When you configure an external KMS provider, `cribl.secret` is removed from disk and managed by that provider instead. Supported providers include AWS KMS and HashiCorp Vault.

Back up `cribl.secret` securely before major changes such as enabling an external KMS or rotating keys.

When using an external KMS, ensure that IAM permissions are correctly scoped (for example, `kms:Decrypt` for AWS KMS). Misconfigured permissions can prevent the Leader from starting. For details, see Configure KMS Providers.

Other Security Files

When you install Cribl Edge, a `keys.json` file is also created. This file contains cryptographic keys used for internal operations within Cribl Edge.

- **Single-instance deployment** - this file is located in `$CRIBL_HOME/local/cribl/auth/`.
- **Distributed deployment** - this file resides on the Leader Node, in `$CRIBL_HOME/groups/<group-name>/local/cribl/auth/`.

Access Secrets

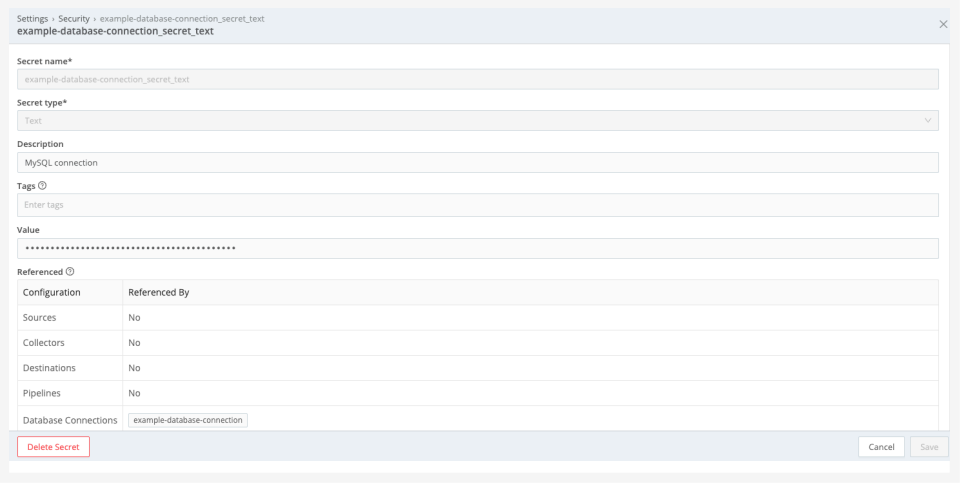
- In a Single-instance deployment, select **Settings** > **Security** > **Secrets**.
- In a Distributed deployment with one Fleet, select **Configure** > **Settings** > **Security** > **Secrets**.
- In a Distributed deployment with multiple Fleets, secrets are managed on each Fleet. Select **Fleet** > `<Name>` > **Fleet Settings** > **Security** > **Secrets**.

On the **Manage Secrets** page, you can configure existing secrets, and/or click **New Secret** to define new secrets.

View Secret References

To view where a secret is used in your deployment:

1. Select a secret from the secrets list.
2. View the **Referenced** table at the end of the secret details. This table lists all Sources, Destinations, Pipelines, or Database Connections that currently reference that secret. If a secret is not referenced in one of those resources, the **Referenced By** column displays **No**.



View where secrets are referenced in your deployment

Add New Secret

The **New Secret** modal provides the following controls:

- **Secret name**: Enter an arbitrary, unique name for this secret.
- **Secret type**: Some options for this field expose additional controls, as described below.
- **Description**: Optionally, enter a description summarizing the purpose of the secret.
- **Tags**: Optionally, enter one or multiple tags related to this secret.

Secret Type

This drop-down offers the following types:

- **Text**: This default type exposes a **Value** field where you directly enter the secret.
- **API key and secret key**: Exposes **API key** and **Secret key** fields, used to retrieve the secret from a secure endpoint. This is the only secret type that Cribl Edge supports for its AWS-based Sources, Collectors, and Destinations, and its Google Cloud Storage Destination.
- **Username with password**: Exposes **Username** and **Password** fields, which you fill to retrieve the secret using Basic Authentication.

Encrypt Secrets Using the Command Line

You also have the option to create and encrypt sensitive values using the same mechanism Cribl Edge employs internally for all sensitive fields. This ensures consistency and leverages the built-in encryption capabilities within Cribl Edge.

This approach allows you to store all your secrets securely in a password store, keeping them out of your Git repository, and adheres to strict organizational security standards that prohibit storing secrets in version control.



Modifying credentials directly through the Cribl Edge UI won't be possible, even in development environments. This approach prioritizes security by keeping secrets out of the UI altogether.

For details, see encrypt in the CLI Reference doc.

Recover cribl.secret

If `cribl.secret` is lost or inaccessible, the Leader will fail to start because it cannot decrypt protected values. Recovery depends on whether you have an unencrypted backup or only an encrypted version.

Recover from an Unencrypted Backup

If you have an unencrypted backup, restore it to the following path and restart the Leader:

```
$CRIBL_HOME/local/cribl/auth/cribl.secret
```

Copy

Recover from an Encrypted Backup (AWS KMS)

If you only have an encrypted version under AWS KMS, use the AWS CLI with `kms:Decrypt` permissions to recover the plaintext key:

```
aws kms decrypt \
  --ciphertext-blob fileb://cribl.secret.encrypted.decoded \
  --region <region> \
  --key-id <KMS key ARN> \
  --output text \
  --query Plaintext | base64 --decode > cribl.secret
```

Copy

Place the recovered file at `$CRIBL_HOME/local/cribl/auth/cribl.secret` and restart the Leader.

Recover After KMS Failure

If your AWS KMS integration is misconfigured and the Leader won't start, switch back to the local KMS provider by editing:

```
$CRIBL_HOME/local/cribl/kms.yml
```

Copy

If you only have the encrypted version `cribl.secret.encrypted`, use the AWS CLI decrypt command above to recover the plaintext key, then restart the Leader.

Rotate cribl.secret

To rotate the master encryption key for the Leader, follow these steps. Note that all credentials encrypted with the current key will need to be re-entered.



Take a full backup of the `$CRIBL_HOME` directory before proceeding. Use `tar` to create an archive and store it in a secure location.

1. Copy the current auth token from the Leader and store it securely. Navigate to **Global Settings** > **System** > **Distributed Settings** > **Leader Settings** > **Auth Token**.
2. Stop the Cribl systemd service.
3. Delete the `$CRIBL_HOME/local/cribl/auth/cribl.secret` file.
4. Edit the `$CRIBL_HOME/local/cribl/auth/users.json` file and reset the password for at least the admin user. For details, see Manual Password Reset.
5. Start the Leader and log in with the new credentials.
6. Restore the auth token you saved earlier. Navigate to **Global Settings** > **System** > **Distributed Settings** > **Leader Settings** > **Auth Token**. Once restored, Edge Nodes should reconnect to the Leader.
7. Reset passwords for other users through the UI, if applicable.
8. Reconfigure TLS and SSO settings if needed.
9. Replace each secret in Sources, Destinations, and Fleets. Secrets are stored in `$CRIBL_HOME/local/cribl/secrets.yml`. To find all references to secrets, search for strings starting with `#42:`:

```
grep -Rn '#42:' $CRIBL_HOME/local/cribl/
```

Copy

10. For each secret, generate a new encrypted value:

```
cribl encrypt -v PLAINTEXT_SECRET
```

Copy

11.4.3. Configure KMS Providers

Cribl Edge's Key Management Service (KMS) maintains the keys that protect sensitive information throughout your deployment. Cribl Edge offers two options for key management:

- **Internal KMS (Stream Internal)** - The default built-in key management system.
- **External KMS Providers** - Enterprise-grade options with HashiCorp Vault and AWS KMS integration (requires Enterprise license).

The table below summarizes the key differences between the Internal vs. External KMS.

Feature	Internal KMS (Stream Internal)	External KMS Providers (HashiCorp Vault and AWS KMS)
Master Key Storage	Stores `cribl.secret` directly on the filesystem.	Stores `cribl.secret` in a separate key management system.
Master Key Location	In a file called `cribl.secret`.	Completely removed from Cribl Stream's filesystem.
License Requirement	Available by default on all license tiers.	Requires an Enterprise license/plan.
Management	Managed entirely within Cribl Edge.	Managed by the external provider (Vault or AWS).
Configuration/Secrets	Maintained through Cribl's Settings interface.	Configured through Cribl's Settings interface.
Setup Complexity	Simpler to set up.	More complex to set up.
Security	Less separation between the application and its encryption keys.	Enhanced security through separation of encryption keys from the application.
Centralized Key Management	None.	Enables centralized key management across your organization.
Availability	Available by default. Unavailable if an external KMS is enabled for the same Leader Node/Fleet.	Unavailable if internal KMS is enabled for the same Leader Node/Fleet.
Supported Providers	N/A	HashiCorp Vault, AWS KMS.

How Encryption Works in Cribl Edge

The Role of `cribl.secret`

At the core of the Cribl Edge encryption system is a master encryption key called `cribl.secret`. This key is used to encrypt sensitive configuration elements such as:

- Authentication credentials in Sources and Destinations (stored in `inputs.yml` and `outputs.yml`)
- API keys and tokens
- Database connection strings (stored in `database-connections.yml`)
- TLS certificates and associated private keys (stored in the `auth/certs` directory
- Other sensitive attributes throughout Cribl Edge
- Secrets stored in `secrets.yml`

KMS Provider Interaction

When using the default Cribl Edge KMS provider:

- The `cribl.secret` value is stored directly on the filesystem in a file called `cribl.secret`.
- Cribl Edge uses this key to encrypt sensitive information in various configuration files.

When you configure an external KMS provider (HashiCorp Vault or AWS KMS):

- The `cribl.secret` file is removed from the filesystem.
- Cribl Edge sets and retrieves the `cribl.secret` value via the chosen external KMS provider.
- The external KMS provider only handles the master encryption key (`cribl.secret`) itself.
- All other sensitive information (Source/Destination credentials, TLS certificates, database connection strings, etc.) remains in the Cribl Edge configuration files, encrypted using the externally-managed key.
- Some sensitive configuration elements may remain in plaintext in the configuration files, depending on the specific attribute.

Manage KMS Across Distributed Environments

In Distributed deployments:

- Each environment (Leader Node, Fleet) can have its own KMS configuration.
- The internal KMS is unavailable for any environment where an external KMS is enabled.

- Other environments without external KMS configured will continue to use the internal KMS.
- Each environment requires its own distinct secret that cannot be shared between environments.

External KMS Providers and Fleets

Some guideines to keep in mind when using an external KMS provider in a Distributed deployment:

- To integrate an external KMS provider into an on-prem Distributed deployment, the Leader Node must have internet access.
- After initial license installation in Distributed mode, you may need to take the following steps to activate KMS features within Fleets:
    - Open **Settings** > **Global** > **Service Processes** > **Processes**.
    - In the list of processes, locate any process with a Role of `CONFIG_HELPER`.
    - Select the **Restart** button next to that process.
    - Upon restarting, the KMS will be available for use in the corresponding Fleet.

Deployment Scenarios

The KMS configuration varies depending on your deployment type:

- In an on-prem Single-instance deployment, you configure the KMS at **Settings** > **Global** > **Security** > **KMS** > **Configure**.
- In a Distributed deployment, you configure the Leader's KMS at the same global location, while additional KMS configs for each Fleet are available at the Fleet's **Group Settings** > **Security** > **KMS** > **Configure** page.

Available KMS Providers

The KMS **Configure** drop-down currently provides these options:

- Stream Internal: The only option available without an Enterprise license/plan. With this option, the secrets themselves are configured and maintained in Cribl Secrets section.
- HashiCorp Vault: Stores the `cribl.secret` key securely in HashiCorp Vault, completely removing it from Cribl's local filesystem.
- AWS KMS: Stores the `cribl.secret` key directly within AWS Key Management Service, completely removing it from Cribl's local filesystem.



External KMS Providers and Fleets

To integrate an external KMS provider into an on-prem Distributed deployment, Cribl Edge's Leader Node must have internet access.

When you initially install a license in Distributed mode, a known bug prevents immediate use of KMS features within Fleets. Here is the workaround:

1. Open **Settings** > **Global** > **Service Processes** > **Processes**.
2. In the list of processes, locate any process with a Role of `CONFIG_HELPER`.
3. Select that process' **Restart** button.

Upon restarting, KMS will be available for use in the corresponding Fleet.

Internal KMS

The **KMS provider** field defaults to `Stream Internal`. With this option, no further configuration here is required (or possible). See Secrets to configure individual secrets.

HashiCorp Vault

Cribl Edge's integration with HashiCorp Vault uses a Vault policy to define the capabilities granted to tokens used for KMS access. This policy ensures tokens have the necessary permissions to interact with Vault paths relevant to KMS functionality.

Vault Policy Permissions

This table outlines the required Vault policy permissions for different access scenarios.

Path	Capabilities	Description
`<path to vault secret>`	Create, Update, Read	Grants access to create, update, and read secrets at the specified path.
`auth/token/create`	Update	Grants permission to update authentication tokens (only required for Token authentication).
`sys/health`	Read	Allows reading system health information (only needed if enabling health checks).

`<path to vault secret>` represents the actual path to the secret within your Vault hierarchy.

Keep these capabilities in mind as you proceed with the configuration.

Setting the **KMS provider** drop-down to `HashiCorp Vault` exposes the following configuration options:

KMS Settings

Vault URL: Enter the Vault server's URL (for example, `http://localhost:8200`).

Namespace: If you are using HashiCorp Vault Enterprise namespaces, enter the desired namespace.

Authentication

Auth Provider: The method for authenticating requests to HashiCorp Vault server. Select one of `Token`, `AWS IAM`, or `AWS EC2`. Your selection determines the remaining **Authentication** options displayed.

Token-based Authentication

Token: Enter the authentication token. This token will be used only to generate child tokens for further authentication actions.

AWS IAM Authentication



In HashiCorp Vault, the term "method" can refer to `userpass`, `token`, or `aws`, among others, but the `aws` method supports two authentication types: `iam` and `ec2`. Meanwhile, in Cribl Edge, you'll see "method" used differently, for example in the **Authentication method** setting described below.

Use the **Authentication method** buttons to select one of the following:

- **Auto**: Uses the AWS instance's metadata service to automatically obtain short-lived credentials from the IAM role attached to an EC2 instance, local credentials, sidecar, or other source. The attached IAM role grants Cribl Edge Edge Nodes access to authorized AWS resources. Can also use the environment variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`. Works only when running on AWS.
- **Manual**: If not running on AWS, you can select this option to enter a static set of user-associated IAM credentials directly or by reference. This is useful for Edge Nodes not in an AWS VPC, for example, those running in a private cloud. It prompts you to provide an **Access key** and a **Secret key**.

Vault AWS IAM Server ID: Value to use for the `Vault-AWS-IAM-Server-ID` header value. This should match the value configured with IAM authentication on Vault.

Vault Role: Authentication role to use in Vault.

Custom Auth Path: If you enabled authentication in HashiCorp Vault with a custom path, enter that path again here.

For example:

- If you enabled authentication with the HashiCorp Vault command `vault auth enable -path /my-auth aws` instead of `vault auth enable aws` you would set a custom path of `my-auth`. Subsequently, when you perform actions using the `vault write` command, you'd specify an auth type with the `auth_type=ec2` or `auth_type=iam` options.

Assume Role

This section is displayed for all AWS IAM authentication methods.

When using **Assume Role** to access resources in a different Region than Cribl Edge, you can target the AWS Security Token Service (STS) endpoint specific to that Region by using the `CRIBL_AWS_STS_REGION` environment variable on your Edge Node. To specify a custom STS endpoint, set the `CRIBL_AWS_STS_ENDPOINT` environment variable. If both are set, `CRIBL_AWS_STS_ENDPOINT` takes precedence. Setting an invalid Region or endpoint results in a fallback to the global STS endpoint.

Enable for Vault Auth: Toggle on if you want to use your Assume Role credentials to access Vault authentication.

AssumeRole ARN: Enter the Amazon Resource Name (ARN) of the role to assume.

External ID: Enter the External ID to use when assuming the role.

Duration (seconds): Duration of the Assumed Role's session, in seconds. Minimum is 900 (15 minutes). Maximum is 43200 (12 hours). Defaults to 3600 (1 hour).

AWS EC2 Authentication

Vault Role: Enter the authentication role to use in Vault.

Custom auth path: If you enabled authentication in HashiCorp Vault with a custom path, enter that path again here. For example:

- You could have used the HashiCorp Vault command `vault auth enable -path /my-auth aws` to enable authentication with a custom path of `my-auth`. Subsequently, when you perform actions using the `vault write` command, you'd specify an auth type with the `auth_type=ec2` or `auth_type=iam` options.

Secret Engine

Mount: Mount point of the Vault secrets engine to use. (Currently, only the KVv2 engine is supported.) Defaults to `secret`.

Secret Path: Enter the path on which the Cribl Edge secret should be stored. For example, `<somePath>/cribl-secret`.

In a Distributed deployment, the Leader, and each Fleet, require a distinct secret. This location cannot be shared between them.

Advanced

Enable Health Check: Whether to perform a health check before migrating secrets data. Defaults to toggled on.

Health check endpoint: Configurable endpoint to use for validating system health. Defaults to `/v1/sys/health`.

AWS KMS

Setting the **KMS provider** drop-down to `AWS KMS` exposes the following configuration options:

Authentication

Authentication method: Select an AWS authentication method.

- **Auto**: This default option uses the environment variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`, or the attached IAM role. Works only when running on AWS.
- **Manual**: You must select this option when not running on AWS.

The **Manual** option exposes these corresponding additional fields:

- **Access key**: Enter your AWS access key. If not present, will fall back to `env.AWS_ACCESS_KEY_ID`, or to the metadata endpoint for IAM role credentials.
- **Secret key**: Enter your AWS secret key. If not present, will fall back to `env.AWS_SECRET_ACCESS_KEY`, or to the metadata endpoint for IAM credentials.

Assume Role

When using **Assume Role** to access resources in a different Region than Cribl Edge, you can target the AWS Security Token Service (STS) endpoint specific to that Region by using the `CRIBL_AWS_STS_REGION` environment variable on your Edge Node. To specify a custom STS endpoint, set the `CRIBL_AWS_STS_ENDPOINT` environment variable. If both are set, `CRIBL_AWS_STS_ENDPOINT` takes precedence. Setting an invalid Region or endpoint results in a fallback to the global STS endpoint.

Enable for KMS: Toggle on if you want to use Assume Role credentials to access the AWS KMS.

AssumeRole ARN: Enter the Amazon Resource Name (ARN) of the role to assume.

External ID: Enter the External ID to use when assuming role. This is required only when assuming a role that requires this ID in order to delegate third-party access. For details, see AWS' documentation.

Duration (seconds): Duration of the Assumed Role's session, in seconds. Minimum is 900 (15 minutes). Maximum is 43200 (12 hours). Defaults to 3600 (1 hour).

Service Configuration

KMS Key ARN: Enter the Amazon Resource Name (ARN) of the AWS KMS Key to use for encryption. This entry is required.

When you configure your IAM account/role in AWS, grant access to the following permissions on the KMS key that will be used:

- `kms:Encrypt`
- `kms:Decrypt`

Then use that account for authentication.

11.5. Compliance

Cribl offers compliance options with the FIPS security standard:

- FIPS on Linux
- FIPS on Windows

For Cribl.Cloud compliance reports and third-party security assessments, see Request Access to TrustCloud.

11.5.1. FIPS Mode For Cribl Edge (Linux)

Federal Information Processing Standards FIPS represent a set of US government standards and guidelines for information security. You can deploy Cribl Edge in **FIPS mode**. This mainly restricts the cryptographic algorithms used within Cribl Edge and also enforces FIPS compliant password requirements.



Do not start Cribl Edge without FIPS mode enabled, or else you will be unable to run it in FIPS mode later. You must enable FIPS mode as described in this section, after installing **but before starting** Cribl Edge.

Requirements

This section describes the system and password requirements for FIPS mode.

FIPS Mode System Requirements

To run Cribl Edge in FIPS mode, your environment must satisfy the following requirements.

Operating Systems

Must support FIPS 140-2 or FIPS 140-3. Several Linux distributions meet this standard. See NIST's list of tested configurations here.



While RHEL 9 can be configured to run in FIPS mode at the operating system level, this **does not automatically enable FIPS mode within Cribl Edge.** You must still follow the procedures outlined in this document to explicitly enable Cribl Edge's FIPS mode. Cribl Edge relies on its own configuration to ensure FIPS compliance, independent of the OS-level FIPS setting.

OpenSSL

The operating system environment must have a FIPS validated version of OpenSSL installed.

- **For Cribl Edge versions 4.5.0 through 4.8.1**: The FIPS provider version must be 3.0.8 or 3.0.9, and the system-wide OpenSSL version must also be 3.0.8 or version 3.0.9, with its associated certificate.
- **For Cribl Edge starting with version 4.8.2**: The FIPS provider version requirement must be >= 3.0.5, and the system-wide OpenSSL version is no longer a factor. Cribl Edge now verifies the OpenSSL version bundled within the binary.

Enterprise Linux 9 is the only version of Enterprise Linux that includes OpenSSL version 3.0.9 out of the box. While it's possible to manually install compatible versions of OpenSSL on older Enterprise Linux versions, this process is complex and generally not recommended. Therefore, the minimum recommended Enterprise Linux version for running Cribl Edge in FIPS mode is 9.

Even when running Cribl Edge on a FIPS-enabled RHEL 9 system, you must still explicitly enable Cribl Edge's FIPS mode as described in the Running Cribl Edge in FIPS Mode section below. The OS being in FIPS mode is not sufficient for Cribl Edge to operate in a FIPS-140 compliant manner.

FIPS Mode Restrictions on Cryptographic Algorithms

When running in FIPS mode, Cribl Edge uses only those cryptographic algorithms that satisfy the FIPS standards. This means that Cribl Edge operates exclusively with FIPS-compliant cryptographic algorithms, disallowing the use of non-compliant algorithms like MD5 (message-digest) or CRC-32 (Cyclic Redundancy Check 32) algorithms.

To ensure proper functionality in FIPS mode, the FIPS provider version must be at least `3.0.5`.

Therefore, in FIPS mode:

Cribl expressions **that rely on MD5 or CRC-32 will fail silently.** This includes expressions that use the `C.Mask.md5()` method and can in turn affect the behavior of parent Functions and their existing parent Pipelines.

The UI will hide certain options normally made available by the typeahead feature.

No Source, Destination, or Email Notification target that uses the Amazon AWS SDK v2 will perform checksums. This is because that SDK uses MD5 to verify checksums, and it applies to both incoming or outgoing payloads. Specifically:

- The Amazon Kinesis Data Streams Source will not perform checksums and the **Verify KPL checksums** setting will not be available.
- The Amazon SNS Notification target will not perform checksums.
- The following Destinations will not perform checksums:
    - Amazon CloudWatch Logs
    - Amazon Kinesis Data Streams
    - Amazon SQS
    - Google Cloud Storage
    - MinIO
    - Prometheus

Role-Based Access Control (RBAC) Controls

Cribl Edge's FIPS mode requires RBAC to be enabled for enhanced security. This ensures proper access management within FIPS-compliant environments. To enable RBAC, the Cribl Edge instance has to be running in Distributed mode. FIPS mode is available on certain license tiers. For details, see Pricing.

If FIPS mode is already enabled and RBAC is subsequently disabled, Cribl Edge will automatically disable FIPS mode and log a message stating:

```
"Disabling FIPS due to Role-based Access Control (RBAC) being disabled."
```

Check `<cribl_install_dir>/local/_system/instance.yml` to ensure Cribl is running in Distributed mode. The default installation directory is typically `/opt/cribl`. The `mode` setting in this file should be set to `distributed`. If it is set to `standalone`, you will need to switch to a Distributed deployment to use FIPS mode with RBAC enabled.

FIPS Mode Password Rules

When Cribl Edge is in FIPS mode, all passwords must:

- Contain eight or more characters.
- Use characters from three or more of the following categories:
  - Lowercase letters.
  - Uppercase letters located after the first character in the password.
  - Digits located before the last character in the password.
  - Non-alphanumeric ASCII characters such as `#`, `!`, or `?`.
  - Non-ASCII characters such as `ñ`, `€`, or emoji.

As of Cribl Edge version 4.5.0, these rules apply for all passwords, whether or not Cribl Edge is running in FIPS mode, with one exception:

- For Cribl Edge not running in FIPS mode, local users whose passwords existed before Cribl Edge 4.4.4 was released can continue to use their passwords, even if those passwords do not satisfy the rules.
- When these users change to new passwords, the new passwords must satisfy the rules.
- New users must create passwords that satisfy the rules.



If you get locked out of your account, you need to reset your password manually.

Running Cribl Edge in FIPS Mode

To begin using Cribl Edge with FIPS enabled, complete either the non-systemd or the systemd procedure below, **before you start Cribl Edge for the first time after installing**.

Enabling FIPS Mode on a Non-systemd Distribution

If the Linux distribution on which you run Cribl Edge **does not** use systemd:

1. Complete **either** of the two following steps:
   - Set the `CRIBL_FIPS` environment variable to `1` (true), **or**,
   - Edit `cribl.yml`, adding this top-level element:

     ```
     fips: true
     ```

     Copy

     Note that `fips: true` is the entire top-level element. For this element, the following "stanza" syntax would be incorrect - **do not** use it:

     ```
     # stanza syntax is incorrect for the fips element
     # even if other elements use it
     fips
       fips: true
     ```

     Copy

2. If you have installed a non-default OpenSSL binary (otherwise, skip this step):
   - Run the following command, and in the output, note the values of `MODULESDIR` and `OPENSSLDIR`.

     ```
     <binary_of_non-default_OpenSSL> version -a
     ```

     Copy

   - Set `OPENSSL_MODULES` to the value of `MODULESDIR`.
   - Set `CRIBL_OPENSSL_DIR` to the value of `OPENSSLDIR`.
3. Ensure RBAC is configured in Cribl Edge.

4. Start Cribl Edge.

Finally, verify that you are in FIPS mode as explained below.

Enabling FIPS Mode on a systemd Distribution

If the Linux distribution on which you run Cribl Edge **does** use systemd:

1. Run the following command, and in the output, note the values of `MODULESDIR` and `OPENSSLDIR`. (As shown, the command will run the default OpenSSL binary. If you have installed a different OpenSSL binary, you can modify the command to run that one instead.)

   ```
   openssl version -a
   ```

   Copy

2. Run the following command, replacing the placeholders with the relevant paths from your environment. (This assumes that you do not already have a `nodejs.cnf` file. If you **do** have one, complete the workaround below instead of this step.)

   ```
   <path_to_Cribl_binary> generateFipsConf -d <path_to_OpenSSL_root_directory>
   ```

   Copy

   If you encounter issues generating the FIPS configuration file, a discrepancy in the OpenSSL path might be the cause. For more detailed troubleshooting specific to RHEL 9, refer to the knowledge base article: Installing Cribl Edge in FIPS Mode on RHEL 9.

   An example of a successful command and its output:

   ```
   useralice@123456791234:/# /opt/cribl/bin/cribl generateFipsConf -d /lib/usr/ssl
   /opt/cribl/state/nodejs.cnf
   ```

   Copy

   This command creates `nodejs.cnf` - the configuration file that NodeJS uses to run in FIPS mode as Cribl Edge starts.

3. Run the following command to open an empty temporary file in an editor:

   ```
   systemctl edit cribl
   ```

Copy

4. Add the following content to the file, replacing the first placeholder with the value of `MODULESDIR` that you noted in Step 1, and the second with the path to your `nodejs.cnf` file:

```
[Service]
Environment="OPENSSL_MODULES=<value_of_MODULESDIR>"
Environment="OPENSSL_CONF=<path_to_nodejs.cnf_file>"
```

Copy

5. Save and exit the editor.
6. Run the following commands to reload the systemctl overrides and start Cribl:

```
systemctl daemon-reload
systemctl start cribl
```

Copy

7. Finally, verify that you are in FIPS mode as explained below.

When a `nodejs.cnf` File Already Exists

If you already have a `nodejs.cnf` file, replace Step 2 above with the following workaround:

1. Create a file with the following content, replacing `<placeholder>` with the value of `OPENSSLDIR`. Name the file something other than `nodejs.cnf`.

```
nodejs_conf = nodejs_init
openssl_conf = nodejs_init

.include /<placeholder>/fipsmodule.cnf

[nodejs_init]
providers = provider_sect

[provider_sect]
default = default_sect
# The fips section name should match the section name inside the
# included fipsmodule.cnf.
fips = fips_sect
[default_sect]
activate = 1
```

Copy

2. Resume the main procedure above, starting with Step 3.

11.5.2. FIPS Mode For Cribl Edge (Windows)

This guide will walk you through the process of setting up and running Cribl Edge in FIPS (Federal Information Processing Standards) mode on Windows systems.

FIPS 140-2 or FIPS 140-3 compliance is required for software deployed in U.S. Federal Government and regulated industries. Cribl Edge supports operation in FIPS mode to meet these requirements.

Prerequisites

- Cribl Edge Windows installer (.msi).
- Administrative access to the Windows system.
- FIPS-validated OpenSSL provider (instructions below).

What You Need to Know About Pre-Configuration



Before configuring FIPS-enabled Edge Nodes, ensure your Leader environment is already configured and running in FIPS mode.

- When running in FIPS mode, Cribl Edge uses only FIPS-approved cryptographic algorithms, ensuring all data handling meets Federal security standards.
- When connecting to a FIPS-configured Leader, Cribl Edge must also be in FIPS mode for full compliance.

Set up Cribl Edge in FIPS Mode

To set up Cribl Edge in FIPS mode, you'll need to build your own OpenSSL FIPS provider. For details, see OpenSSL: Notes for Windows platforms.

Build Your Own FIPS Provider

This section guides you through the process of building your own OpenSSL FIPS provider.

Install Required Tools

1. Install Strawberry Perl, which includes the Netwide Assembler – (NASM).
2. Microsoft C++ toolchain (Visual Studio).
3. If you have `git` installed, you may need to rename/move its `perl.exe` to avoid conflicts.

Build the Provider

1. Find and open the Visual Studio (20XX) Native Tools Command Prompt for `x64` architecture. You can search for it, by typing `x64` in the **Start** menu. Right-click on it and select **Run as administrator**. (Do not use PowerShell or plain CMD).
2. Download the latest FIPS-validated version of OpenSSL:

   ```
   curl -LO https://www.openssl.org/source/openssl-3.0.9.tar.gz
   tar -xf openssl-3.0.9.tar.gz
   cd openssl-3.0.9
   ```

   Copy

   If you run into an error message: `ERROR: module machine type 'x86' conflicts with target machine type 'x64'`, it means you're not using the right version of the command prompt for what you're trying to do. Make sure you've selected the `x64 Native Tools Command Prompt`.
3. Run the following commands in the Visual Studio Developer Command Prompt (with `x64` architecture selected):

   ```
   perl configure VC-WIN64A enable-fips
   nmake
   ```

   Copy

   The build process might take considerable time to complete. Once the build is complete, you'll have a `providers` directory. The providers directory is created within the OpenSSL build directory (for example, `openssl-3.0.9/providers/`).
4. Once built, you'll need the following files from the build directory:
   - `providers\fipsmodule.cnf` - This specific file from your build is essential as it contains a unique hash that validates the DLL you built.
   - `providers\fips.dll`
   - `providers\legacy.dll`

Set up the Provider

This step creates a dedicated directory for Cribl's FIPS-related files and copies the necessary DLL and configuration files into it, preparing the application for FIPS-compliant operation.

In the same Native Tools Command Prompt, type the following:

```
mkdir "C:\Program Files\Cribl-FIPS"
copy providers\fips.dll "C:\Program Files\Cribl-FIPS"
copy providers\legacy.dll "C:\Program Files\Cribl-FIPS"
copy providers\fipsmodule.cnf "C:\Program Files\Cribl-FIPS"
```

Copy

Generate the `nodejs.cnf` Configuration File

This step creates a necessary configuration file that enables FIPS compliance for the `Node.js` environment used by Cribl. You have two options.

Option 1: Manual Configuration Creation

1. Open an Administrative Command Prompt
2. Run the following command to create and open the configuration file:

```
notepad "C:\Program Files\Cribl-FIPS\nodejs.cnf"
```

Copy

3. Copy and paste the following configuration content into the file:

```
nodejs_conf = nodejs_init
openssl_conf = nodejs_init
.include "C:\\Program Files\\Cribl-FIPS\\fipsmodule.cnf"

[nodejs_init]
providers = provider_sect
alg_section = algorithm_sect

[provider_sect]
default = default_sect
fips = fips_sect

[default_sect]
activate = 1

[algorithm_sect]
default_properties = fips=yes
```

Copy

4. Save the file and close Notepad.

This configuration redirects `Node.js` to use FIPS-compliant cryptographic modules and algorithms. The file includes references to the FIPS module configuration and sets default properties to enforce FIPS compliance across all cryptographic operations.

Option 2: Using the Cribl CLI (Post-Installation)

Alternatively, after installing Cribl Edge, you can generate the configuration file using the Cribl CLI:

```
cribl generateFipsConf -d <fully qualified path to fipsmodule.cnf>
```

You can generate the `nodejs.cnf` configuration once on any machine and then distribute it to others, eliminating the need to repeat this step on each individual machine.

Install Edge with FIPS Configuration

To install Cribl Edge with FIPS mode enabled:

1. Use the `SETENVVARS` parameter with the MSI installer:

```
msiexec /i CriblEdge-<version>.msi SETENVVARS="OPENSSL_MODULES=C:\Program Files\Cribl
```

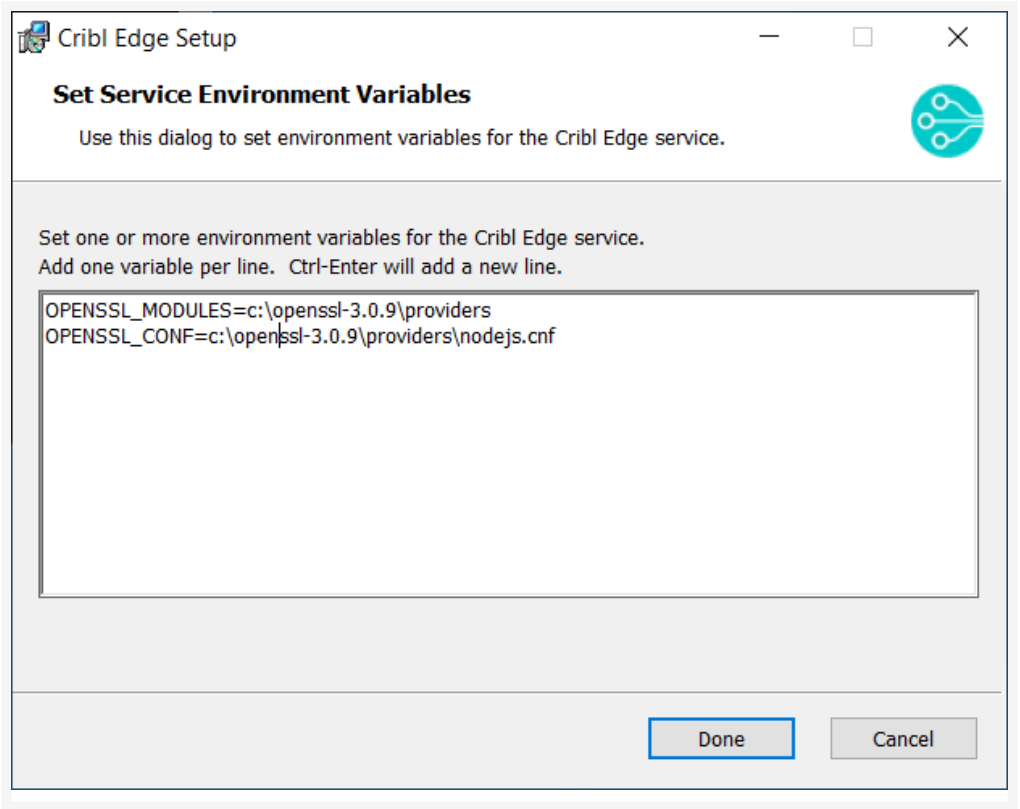
Copy

2. This opens the Windows Installer Wizard. Choose `Managed Mode` and complete the configuration to connect to the FIPS-enabled Leader.

When executing the MSI installer command, pay attention to these important parameter requirements:

- The pipe character (|) separates multiple environment variables.
- Make sure to enclose the entire `SETENVVARS` string in quotes.
- `OPENSSL_MODULES` points to the directory containing the provider files.
- `OPENSSL_CONF` points to the specific configuration file.

Alternatively, you can set the environment variables through the Installation Wizard:



Set Environment Variables

For details, see Installing Cribl Edge on Windows.

Confirm Cribl Edge is in FIPS Mode

After installation, validate that Edge is running in FIPS mode by checking for FIPS initialization messages:

1. On the Leader or Edge Node, search `cribl.log` for this message:

```
"level":"info","message":"running with FIPS enabled", "OpenSSLVersion":"3.0.13+quic
```

Copy

2. In the UI, navigate to: **Node Settings** > **Diagnostics** > **System Info**.

The presence of the above message confirms that Cribl Edge is in FIPS mode.

Troubleshooting

11.5.3. SELinux (Enforcing Mode) Configuration

SELinux (Security-Enhanced Linux) is a mandatory access control system for Linux that provides a granular level of security by defining access controls for applications, processes, and files. It uses security policies, which are a set of rules that tell SELinux what can or can't be accessed, to enforce the access allowed by a policy.

SELinux has two modes, **enforcing** and **permissive**. When running in enforcing mode, it strictly enforces the SELinux policy and denies access based on policy rules. This can sometimes cause conflicts with applications that aren't explicitly allowed to perform certain actions.

Cribl Edge and SELinux

To effectively integrate Cribl Edge with SELinux, consider the different policy requirements for in-stream processing and network ports.

- Cribl Edge's in-stream processing often involves predefined processes and files, allowing for a relatively restrictive SELinux policy.
- Network ports might require a more permissive policy because users configure them within the UI.

Key Steps for SELinux Configuration

To ensure optimal integration of Cribl Edge with SELinux, begin by verifying the current SELinux status and the context of the Cribl Edge binary.

Check SELinux Status

Use the `getenforce` command to determine the current status of SELinux on your system.

The output will be one of the following:

- `Enforcing`
- `Permissive`
- `Disabled`

If the output is `Enforcing`, SELinux is enabled and enforcing policies.

Verify Cribl Edge's SELinux Context

Use the command `ls -Z /opt/cribl/bin/cribl` to list the contents of the directory `/opt/cribl/bin` while displaying their SELinux security contexts.

The output should show the SELinux context for the Cribl Edge binary.

Example output:

```
-rwxr-xr-x. cribl cribl cribl unconfined /opt/cribl/bin/cribl
```



Deploy Cribl Edge directly to `/opt` to avoid context issues. Do not deploy Cribl Edge to a user's home directory, or to root (`/`), and then move it to `/opt`.

Revert Labels on Files and Directories

If needed, you can also use `restorecon`, an SELinux policy utility from the `policycoreutils-python-utils` package, to revert labels on files and directories to their default SELinux security context:

```
restorecon -R -v /opt/cribl
```

Copy

Check for SELinux Denials

Use `auditd` to log SELinux denials. Look for messages related to Cribl Edge in the `/var/log/audit/audit.log` file.

Modify SELinux Policy

If you encounter SELinux denials, you might need to modify the SELinux policy. There are three options ranging from least secure to most secure.

Use `semanage` (Less Secure)

Using `semanage` is the simplest but least secure method. It grants permissive access to the `cribl_t type`, allowing all operations.

```
semanage permissive -a cribl_t
```

Copy

Use `audit2allow` (Slightly More Secure)

Using `audit2allow` creates SELinux policy rules to permit actions that were previously denied.

To identify and address SELinux denials related to Cribl Edge, follow these steps:

1. Find denied operations:

```
sudo grep cribl /var/log/audit/audit.log | audit2allow -M cribl
```

Copy

2. Install the generated policy:

```
sudo semodule -i cribl.pp
```

Copy

Create a Custom Policy (Most Secure)

Use tools like `secil` or `policycoreutils` to create a custom policy tailored to Cribl Edge's needs. This provides the highest level of security but requires more technical expertise.

Reload SELinux Policy

Once you've altered the SELinux policy using tools like `semanage`, `audit2allow`, or through custom policy creation, use this command to ensure that the new rules take effect immediately.

```
sudo semanage reload -a
```

Copy

Specific Cribl Edge Considerations

In addition to the general SELinux guidelines, pay close attention to these Cribl Edge-specific factors:

- **Cribl Edge configuration**: Review Cribl Edge configuration files for any settings that might conflict with SELinux restrictions.
- **Integration with other applications**: Ensure that Cribl Edge's interactions with other applications are compatible with their SELinux contexts and permissions.

Additional Considerations

In addition to the basic configuration, these factors are also important for optimal performance:

- **SELinux booleans**: Check the SELinux documentation for relevant booleans that might affect Cribl Edge's behavior.
- **SELinux context labeling**: Ensure files and directories used by Cribl Edge have the correct SELinux context. Use tools like `restorecon` to relabel files.
- **SELinux troubleshooting tools**: Use tools like `audiscan` and `audit2why` to analyze SELinux audit logs.

SELinux Troubleshooting

SELinux problems usually surface when you try to start the Cribl Edge service. Users immediately receive an error like the following:

```
Job for cribl.service failed because the control process exited with error code.
See "systemctl status cribl.service" and "journalctl -xe" for details.
```

Copy

One indication that the problem is in SELinux configuration, not in the Cribl Edge service itself, is when (1) nothing has been logged in `$CRIBL_HOME/log/cribl.log`, and (2), when you run `journalctl -xe` you see errors like the following:

Errors from journalctl -xe

```
cribl.service: Failed to execute command: Permission denied
cribl.service: Failed at step EXEC spawning /opt/cribl/bin/cribl: Permission denied

cribl.service: Control process exited, code=exited status=203
cribl.service: Failed to execute command: No such file or directory
cribl.service: Failed at step EXEC spawning /bin/rm -f /opt/cribl/pid/cribl.pid: Perm
```

Copy

You can confirm that SELinux configuration is the issue by verifying that the Cribl Edge service is able to start up on its own. To do this, run the `cribl start` command.

Here are three commands that you can think of as a basic SELinux troubleshooting toolkit:

- `getenforce` verifies that SELinux is enabled and enforcing policy.

- `sestatus -v` shows details of the SELinux policy that's in effect.
- `ls` with the `-Z` or `--context` option shows the SELinux tags on files and/or folders. In the output, the fifth column reports the SELinux context in the form `user:role:type:level`.

Example Problem

Suppose you run `ls -lZ /opt/cribl` to see the SElinux context for `/opt/cribl`.

If the output looks like the following, there is a problem: the presence of "home" among the `type` part of the Linux context. In this case, `user_home_t` is the offending type. When the type is `user_home_t` or `admin_home_t`, SELinux will not permit the service to read, write, or execute files.

Problematic SELinux context output

```
$ ls -laZ /opt/cribl
total 0
drwxr-xr-x. 6 cribl cribl unconfined_u:object_r:user_home_t:s0  62 Feb 16 08:31 .
drwxr-xr-x. 3 root  root  system_u:object_r:usr_t:s0            19 Feb 16 19:54 ..
drwxr-xr-x. 2 cribl cribl unconfined_u:object_r:user_home_t:s0 202 Feb 16 08:31 bin
drwxr-xr-x. 5 cribl cribl unconfined_u:object_r:user_home_t:s0  49 Feb 16 08:20 data
drwxr-xr-x. 6 cribl cribl unconfined_u:object_r:user_home_t:s0  59 Feb 16 08:31 defau
drwxr-xr-x. 3 cribl cribl unconfined_u:object_r:user_home_t:s0  22 Feb 16 08:31 third
```

Copy

Example Solution

To fix the labeling, download Cribl Edge again directly to `/opt`. **Do not** download Cribl Edge to a user's home directory, or to root (`/`), and then move it to `/opt` - that was what caused our example problem.

Downloading Cribl to /opt

```
curl -Lso - $(curl -s https://cdn.cribl.io/dl/latest) | sudo tar zxvf - -C /opt
```

Copy

A simpler option is to revert the labels using `restorecon`:

Reverting labels with restorecon

```
restorecon -R -v /opt/cribl
systemctl restart cribl
```

Copy

Once you correct the problem, the output will look like the following. Instead of `*home_t` listings in the `type` field, this SELinux context has `usr_t`, a type that will satisfy SELinux's criteria for allowing the service to read, write, and execute files.

Corrected SELinux context output

```
# ls -laZ /opt/cribl
total 0
drwxr-xr-x. 6 cribl cribl unconfined_u:object_r:usr_t:s0  62 Feb 16 08:31 .
drwxr-xr-x. 3 root  root  system_u:object_r:usr_t:s0      19 Feb 16 20:05 ..
drwxr-xr-x. 2 cribl cribl unconfined_u:object_r:usr_t:s0 202 Feb 16 08:31 bin
drwxr-xr-x. 5 cribl cribl unconfined_u:object_r:usr_t:s0  49 Feb 16 08:20 data
drwxr-xr-x. 6 cribl cribl unconfined_u:object_r:usr_t:s0  59 Feb 16 08:31 default
drwxr-xr-x. 3 cribl cribl unconfined_u:object_r:usr_t:s0  22 Feb 16 08:31 thirdparty
```

Copy

12. Integrations

Integrations enable seamless data flow between various systems. They encompass both Sources and Destinations, allowing you to get data from multiple origins and deliver processed data to various endpoints. Integrations are the entry and exit points in the event processing order.

- Sources are configurations that enable Cribl to receive data from various origins. They serve as the entry points in the event processing order, enabling the collection of raw data for subsequent processing.
- Destinations are configurations that define where Cribl sends processed data. They act as the exit points in the event processing order, sending enriched and transformed events to the appropriate systems for storage, analysis, or further processing.

Key Concepts for Optimal Data Flow

- Destination Backpressure Triggers: Understanding the conditions that can cause backpressure in Destinations is crucial for preventing data loss and ensuring smooth data flow, especially during peak load periods.
- Persistent Queues: To safeguard against data loss when integrations are slow or unavailable (backpressure events), persistent queues provide a temporary storage mechanism on disk.
- Load Balancing: Distributing data evenly across multiple Sources and Destinations is essential for maximizing performance and reliability.

12.1. Common Data Flows

The following topics describe common data flows in Cribl Edge.

Topic	Description
Collect Logs with File Monitor	Collect logs from a variety of files on your systems using the File Monitor
Collect Windows Event Logs and Metrics	Gain real-time visibility into system health, security, applications, and performance of your Windows environment by collecting Windows Event Logs and Windows metrics with Cribl Edge
Cribl Edge to Cribl Stream	Sending data from Cribl Edge to Cribl Stream

12.1.1. Collect Logs With File Monitor

Collect logs from a variety of files on your systems using the File Monitor.

To collect logs from files (including text files, compressed, archived, and some binary log files), you can use the File Monitor.



For a full description and reference of the Source's settings, see File Monitor Source.

The File Monitor Source collects those logs and converts them to events based on the extracted lines and records.

Cribl Edge comes with two preconfigured, but disabled File Monitor Sources: `in_file_auto` and `in_file_varlog`.

`in_file_auto` uses the **Auto** discovery mode, which automatically discovers files that running processes have open for writing. `in_file_varlog` is configured with the **Manual** discovery mode, which enables you to specify a single path from which Edge will collect log files. The predefined Source is aimed at the `/var/log/` path.

Use Preconfigured File Monitor Sources

To use one of the preconfigured File Monitor Sources:

1. On the top bar, select **Products**, and then select **Edge**. Under **Fleets**, select a Fleet.
2. Select **Collect**.
3. Select the **File Monitor** stack to view both preconfigured Sources.
4. Select and drag the plus icon next to one of the Sources to a Destination (for starters, you can choose DevNull).
5. In the **Connection Configuration** modal, select how you want the data to flow through to the Destination. Select `Passthru` to send all the data without processing, then select **Save**.
6. To start the Source, select **Commit**, enter a commit name, and select **Commit & Deploy**.

Now, ensure that data is being collected: hover over the File Monitor Source and select **Capture**. After a short time you should see sample data captured by the Source. Once you're satisfied, you can switch from DevNull to your desired target Destination.

Monitor Files in Specific Folders

To manually select a specific path to discover files in, configure a new File Monitor Source.

1. On the top bar, select **Products**, and then select **Edge**. Under **Fleets**, select a Fleet.
2. Select **More > Sources**. Select **File Manager**.
3. Next, select **Add Source**.
4. Enter a unique identifier for the Source in **Input ID**.
5. Make sure **Discovery Mode** is set to **Manual**.
6. In **Search path**, provide the path to the directory you want to discover. The path can't contain wildcards.
7. Optionally, set **Max depth** to define whether collection should be limited to a specified depth of subdirectories. To limit collection to the specified **Search path** only, set **Max depth** to `0`. A value of `1` will include only the next level of subdirectories, and higher values will dig deeper.

Further settings allow you to fine-tune the log collection.

You can control which files are monitored by filling in the **Filename allowlist**. For guidance on how to use allowlists without creating duplicates, see Using Allowlists Effectively.

To make sure that all desired files are read, you can fine-tune the length and composition of hashes that the File Monitor uses to track files. See File Tracking with Hashes for more information.

12.1.2. Collect Windows Event Logs And Metrics

Collecting Windows Event Logs and metrics in Cribl Edge allows you to gain real-time visibility into system health, security, applications, and performance of your Windows environment.

Collect Windows Event Logs

To collect Windows Event Logs, you need to create and configure a Windows Event Log Source and point it at correct logs on the server.



For a full description and reference of the Source's settings, see Windows Event Logs Source.

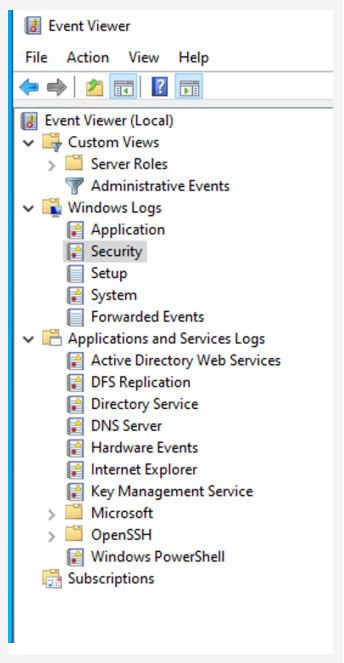
Locate Windows Event Logs on the Server

Event logs in Windows Servers are classified into the following broad categories:

- **System**: Events related to the system and its components. Failure to load the boot-start driver is an example of a system-level event.
- **Applications and Services**: Events related to software or an application hosted on a Windows computer get logged under the application event log. For example, if a user encounters a problem in loading the app, it will be logged.
- **Security**: Events related to the safety of the system. The event gets recorded via the Windows auditing process. Examples include failed authentication and valid logins.
- **Setup**: Events occur during the installation of the Windows operating system. On domain controllers, this log will also record events related to Active Directory.
- **Forwarded Events**: Contains event logs forwarded from other computers in the same network.

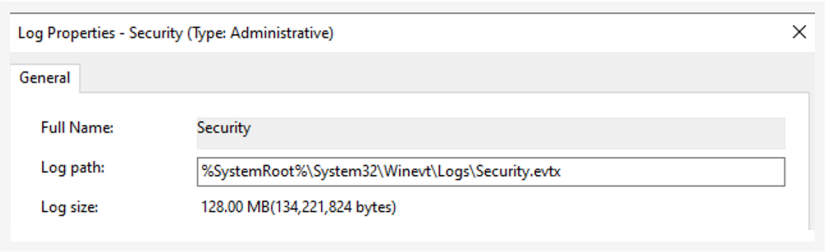
To find out what other event logs might be of interest:

1. Go to your Windows Server Event Viewer.



Event Viewer

2. To get the accurate name of a given event log, right-click it and select **Properties**.
3. From the **Properties** modal, copy the **Full Name** value to your clipboard. You'll use this value when configuring a Windows Event Log Source.

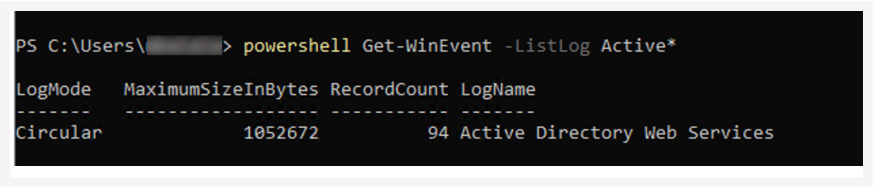


Log properties: Full name from the UI

Alternatively, you can execute the following PowerShell command to list a particular log's name. In this example, we are using a `*` wildcard.

```
powershell Get-WinEvent -ListLog Hardware*
```

Copy



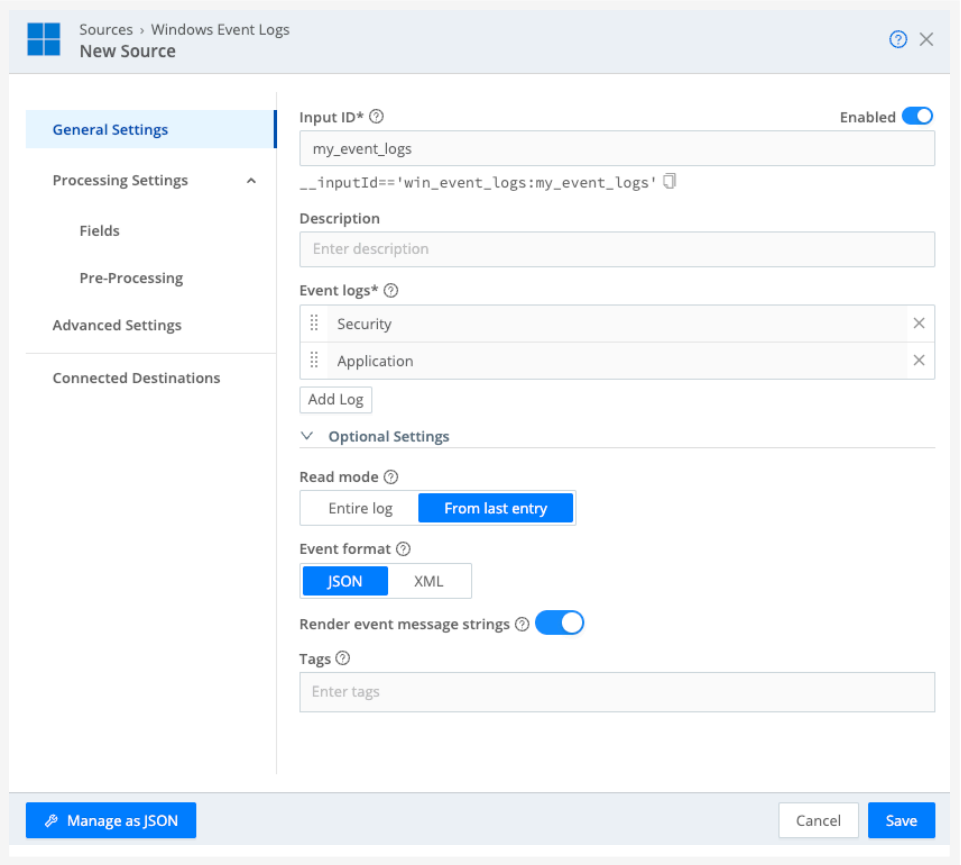
Log Name from PowerShell

From the Powershell output, the **LogName** field is what you would specify in the Cribl UI. In the above screenshot, for example, you would enter `Active Directory Web Services` in the Cribl UI.

Configure the Windows Event Logs Source

Once you have the accurate names of the event logs you want to collect, you can configure and enable the Windows Event Logs Source in Cribl Edge.

1. On the top bar, select **Products**, and then select **Edge**. Under **Fleets**, select a Fleet.
2. Select **More** > **Sources** (Edge). Select **Windows Event Logs**
3. Next, select **Add Source**.
4. Enter a unique identifier for the Source in **Input ID**.
5. In **Event logs**, enter the log name you copied from the Event Viewer.
6. Optionally, configure other Source settings.



Configuring Windows Event Logs

7. Save the Source configuration.
8. To start the Source, select **Commit**, enter a commit name, and select **Commit & Deploy**.

After a few moment, you can check the **Live Data** tab and capture some data to make sure the logs are generated.

You can now send the event logs to your visualization tool of choice or any of Cribl Edge's supported Destinations.

Collect Windows Metrics

To collect Windows metrics rather than log events, configure Cribl Edge's Windows Metrics Source.

For a full description and reference of the Source's settings, see Windows Metrics Source.

A Windows Metrics Source is preconfigured for each of your Fleets with basic settings and enabled automatically. To have further control over the collection, modify its configuration. In particular, decide on the level of granularity for the metrics you want to collect.

To do this, in the **Windows Metrics** Source modal, select **Host Metrics**, then **Custom**. Here, we recommend setting the following granularity options for three of the configurable types:

- **CPU**: Select **Custom**, then enable **Per-CPU metrics**.
- **Network**: Select **Custom**, then enable **Per interface metrics**.
- **Disk**: Select **Custom**, then enable **Per volume metrics**.

When done, remember to commit and deploy your changes.

12.1.3. Collect Kubernetes Logs, Events, And Metrics

Collect Kubernetes data with Cribl Edge to forward them for further observability and analytics.

Kubernetes environments generate large volumes of observability data, including logs, events, and metrics, making data collection complex. Cribl Edge offers the Kubernetes Logs, Kubernetes Events, and Kubernetes Metrics Source, as well as Prometheus Edge Scraper to simplify this task.

The Kubernetes Logs, Kubernetes Metrics, and Prometheus Edge Scraper Sources support disk spooling, which allows Cribl Search to query their output.

Collect Kubernetes Logs

The Kubernetes Logs Source connects to the Kubernetes API and loads the lists of Pods on the node on a configurable polling interval.



For a full description and reference of the Source's settings, see Kubernetes Logs Source.

Enable Preconfigured Kubernetes Logs Source

Cribl Edge has a preconfigured **in_kube_logs** Source that you can adapt to your needs. To enable log collection with the Source:

1. On the top bar, select **Products**, and then select **Edge**. Under **Fleets**, select a Fleet.
2. Select **Collect**.
3. Select the **in_kube_logs** Kubernetes Logs Source.
4. Select and drag the plus icon next to it to a Destination (while testing the procedure, you can choose DevNull).
5. In the **Connection Configuration** modal, select how you want the data to flow through to the Destination. Select `Passthru` to send all the data without processing, then select **Save**.
6. To start the Source, select **Commit**, enter a commit name, and select **Commit & Deploy**.

Now, ensure that data is being collected: hover over the Kubernetes Logs Source and select **Capture**. After a short time you should see sample data captured by the Source. Once you're satisfied, you can switch from DevNull to your desired target Destination.

Finetune Kubernetes Log Collection

Now that you have the basic Kubernetes log flow configured, you can customize the Source settings.

Configure Log Polling Interval

Set the polling interval based on the expected start and stop intervals of Pods in your environment. Higher volume log sources may require a faster polling interval to avoid rotating logs faster than you can collect them.

For example, if the container emits `1 MBps` and rotates logs at `10 MB`, the polling interval must be less than `10s`, to prevent hitting the `10 MB` rotation threshold twice in 10 seconds.

For more information about configuring the polling interval, see Polling Interval and Log Rotation.

Configure Log Filter Rules

The Source determines which Pods to report on based on the rules defined in **Filter Rules**. The default `!metadata.namespace.startsWith('kube-')` rule means the Source ignores Pods in the `kube-*` namespace.

You can use any Kubernetes Pod object fields in the filter, such as `metadata`, `spec`, and `status`.

Below the **Capture** button, select the Advanced Settings ⋯ button and then toggle **Show Internal Fields** on to explore the `__metadata` fields forwarded from the Cribl Edge Nodes. These metadata fields show information about the Pod, Node, and Cribl environment where they were collected.



See Kubernetes Pod Object definition for details.

Collect Kubernetes Events

Kubernetes events include state changes (such as Pod starts and jobs that fail to start) occurring on the Kubernetes cluster regarding workloads. To monitor such events, use the preconfigured Kubernetes Events Source called **in_kube_events**.

1. On the top bar, select **Products**, and then select **Edge**. Under **Fleets**, select a Fleet.
2. Select **Collect**.
3. Select the **in_kube_events** Kubernetes Events Source.
4. Select and drag the plus icon next to it to a Destination (while testing the procedure, you can choose DevNull).

5. In the **Connection Configuration** modal, select how you want the data to flow through to the Destination. Select `Passthru` to send all the data without processing, then select **Save**.
6. To start the Source, select **Commit**, enter a commit name, and select **Commit & Deploy**.

Now, ensure that data is being collected: hover over the Kubernetes Events Source and select **Capture**. After a short time you should see sample data captured by the Source. Once you're satisfied, you can switch from DevNull to your desired target Destination.

For a full description and reference of the Source's settings, see Kubernetes Events Source.

Configure Event Filter Rules

The Source determines which events to collect based on the rules defined in **Filter Rules**. No default rules are defined, which means all emitted events are collected.

As an example, you can filter out events based on their `type` with `object.type.equals('Warning')` or, based on the `reason`, with `object.reason.equals('Failed')`.

Collect Kubernetes Metrics

The Kubernetes Metrics Source lets you generate metrics periodically based on the status and configuration of a Kubernetes cluster and its Nodes, Pods, and containers.

See a full description and reference of the Source's settings in Kubernetes Logs Source, and more information about metrics collected in Kubernetes Metrics Details.

Collect Metrics with Prometheus Edge Scraper

You can also configure the Prometheus Edge Scraper to collect metrics from Pods and Nodes (as well as other Prometheus targets).

To collect Kubernetes metrics, Cribl Edge uses the `/metrics` and `/metrics/cadvisor` endpoints on the kubelet. This information gives you a good idea of what's going on on each Node and Pod.

Using the Prometheus Edge Scraper, you can discover and collect metrics in Kubernetes Clusters dynamically. There are two Kubernetes-specific modes of operation:

- **Kubernetes Node**: When configured to scrape Nodes, each Cribl Edge Node (Pod) will scrape from the configured port on every deployed Node in the DaemonSet. Use this to scrape endpoints on the Kubelet and other node-local interfaces.

- **Kubernetes Pods**: When configured for Pod scraping, a single Edge Node will be elected to scrape the metrics from the discovered endpoints.

For a full description of how to configure the Source and a reference of its settings, see Prometheus Edge Scraper.

Goat-to Resources

Monitoring Kubernetes with Edge — Video

Managing Kubernetes Events with Cribl Edge — Blog

12.1.4. Cribl Edge To Cribl Stream

Cribl Edge automatically discovers logs, metrics, application data, and so on - in real time - from your configured endpoints, and delivers them to Cribl Stream or any supported destination. Meanwhile, Cribl Stream can help collect, reduce, enrich, transform, and route data from Cribl Edge to any destination. And using a Cribl HTTP Source, you can collect and route data from Edge Nodes to Stream Worker Nodes without incurring additional cost.

For more information about sending data between organizations and Workspaces without paying for data twice, see Transfer Data Between Workspaces or Environments.

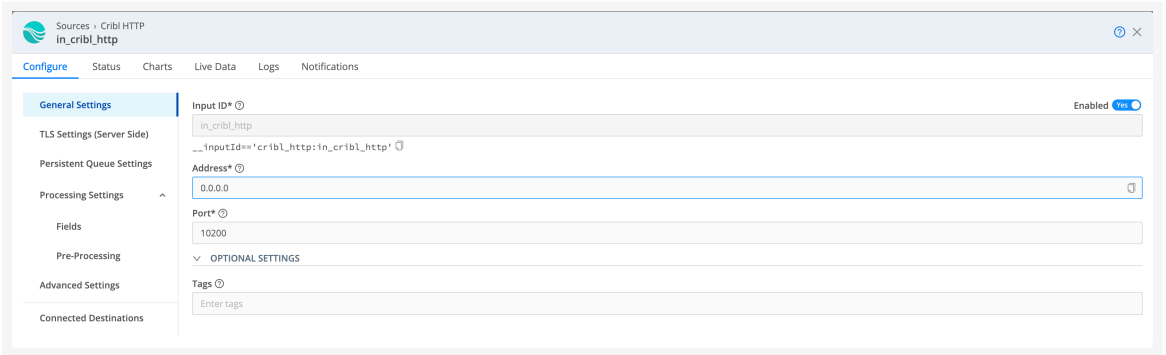
This guide outlines how to route data from an Edge Node (or an entire Fleet) to an existing Stream Worker Group for additional processing. We will walk you through the following:

- Configure Cribl HTTP Source on Cribl Stream to receive data from the Edge Node.
- Configure the Exec Source on Cribl Edge to collect data on the Edge Node.
- Configure the Cribl HTTP Destination on Cribl Edge to send data to Cribl Stream.
- Configure a Route to Send the Data.

And finally, we will confirm the data flow.

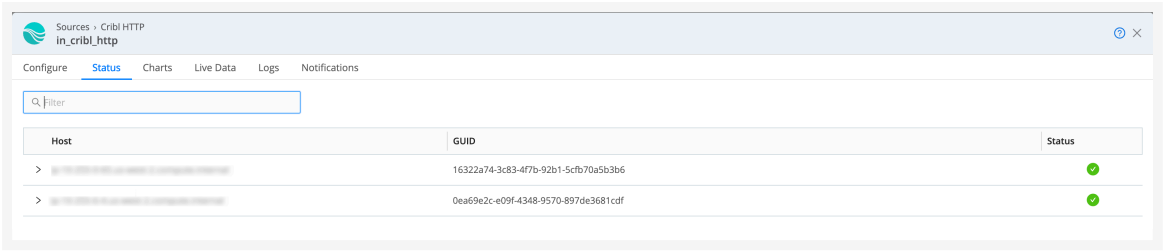
Configure the Cribl HTTP Source on Cribl Stream

In Cribl Stream, start by configuring and enabling a Cribl HTTP Source. The key requirement here is to set the **Port** to listen on. For this example, we'll use the default `10200`. (Optionally, you can also configure TLS, enable persistent queue, and/or a pre-processing Pipeline.)



Configuring a Cribl HTTP Source

When done, **Commit** and **Deploy** your changes. Before moving on to the next step, confirm that your Source is healthy.



Status of the Cribl HTTP Source

On Cribl-managed Workers in Cribl.Cloud, make sure that TLS is either disabled on both the Cribl HTTP Source and the Cribl HTTP Destination it's receiving data from, or enabled on both. Otherwise, no data will flow. In the Source, TLS is enabled by default.

Configure the Exec Source on Cribl Edge

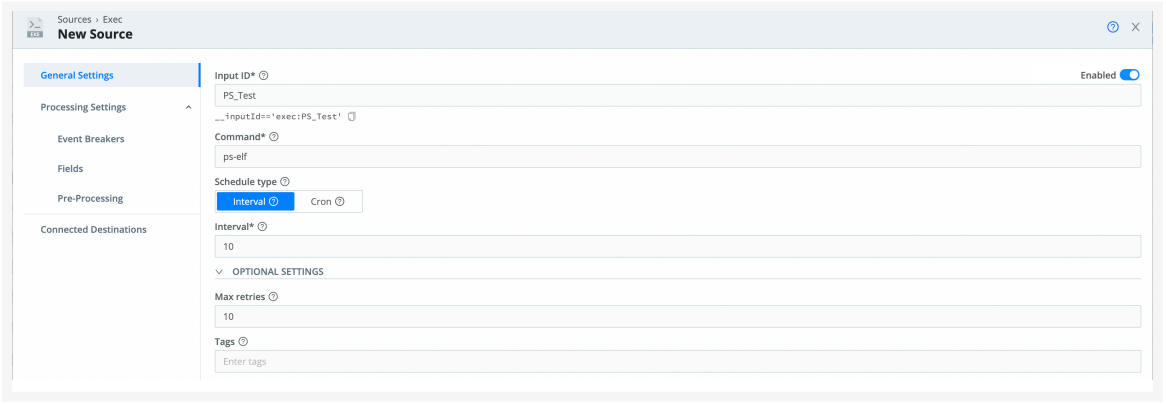
Next, we'll configure the Exec Source on your Edge Node. This Source will break the incoming streams of data into discrete events, and send them to Cribl Stream.

In this step, you can swap out the Exec Source by instead configuring a System Metrics or File Monitor Source. Or, configure multiple Sources to connect to the same Destination.

The Exec Source enables you to periodically execute a command and collect its `stdout` output. In the Exec Source's configuration modal, specify:

- Which command to execute.
- The number of times to attempt running the command.
- The interval between attempts.

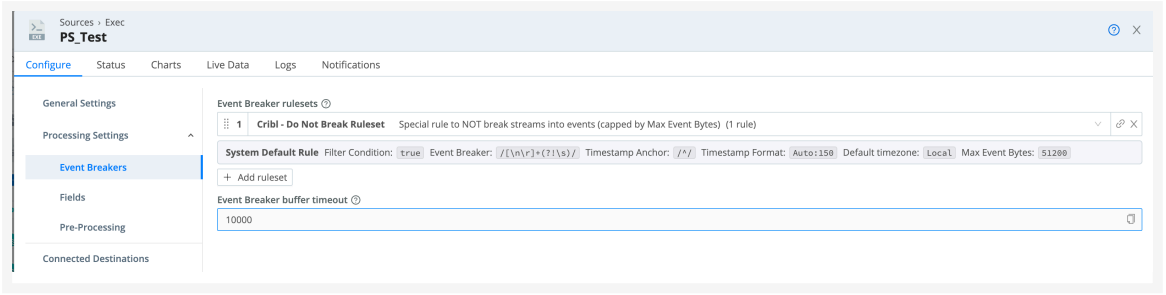
In our example, we are running the `ps` command to list and retrieve running processes every `10` seconds.



Configuring an Exec Source

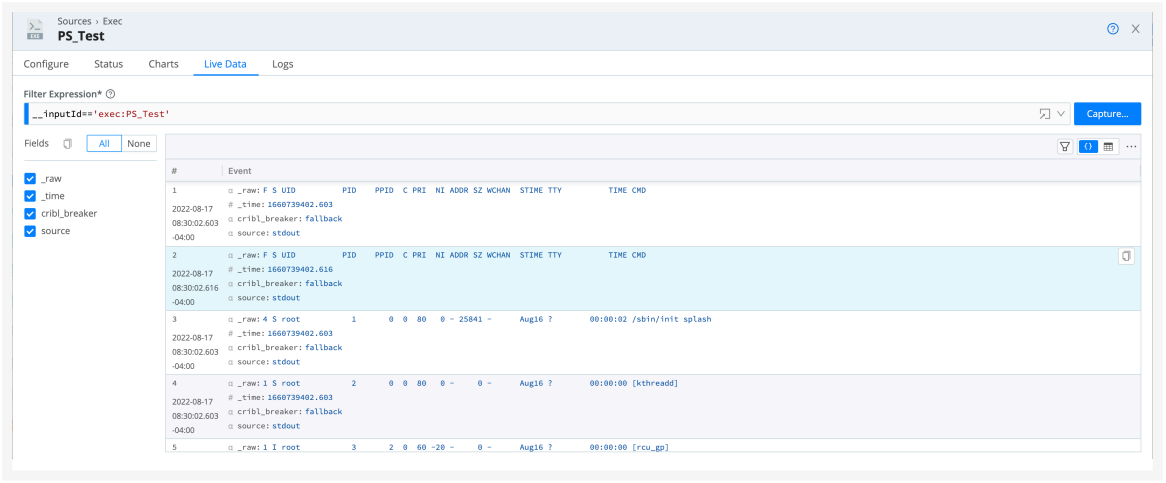
If we don't configure an Event Breaker, then with each capture we run on the dataset, each process will be ingested as its own event, without the header information. So to structure the data, we'll add an Event Breaker.

On the Exec Source configuration modal's left tab, select **Event Breaker**. In the **Event Breaker rulesets** drop-down, select `Cribl - Do Not Break Ruleset`.



Apply an Event Breaker

Next, preview your data on the modal's **Live Data** tab.



Preview Live Data

Configure the Cribl HTTP Destination on Cribl Edge

To get the data flowing, we'll configure the Cribl HTTP Destination on your Edge Node. A few things to note when configuring this Destination:

- The Cribl endpoint field in this Destination must point to the **Address** and **Port** you entered on the Cribl HTTP Source you configured in the first step.
- Optionally, define your **Compression**, **Throttling**, and **Backpressure behavior** requirements.
- On Cribl-managed Worker/Edge Nodes in Cribl.Cloud, make sure that TLS is either disabled on both the Cribl TCP Destination and the Cribl TCP Source it's sending data to, or enabled on both. Otherwise, no data will flow. In the Destination, TLS is disabled by default.

Once you've configured your Destination, test it to verify that your Edge Node can communicate with the Stream Worker Group.

Configure a Route to Send the Data

Finally, configure a Route to send your data to Cribl Stream. For this example, we can use the `passthru` Pipeline.

Confirm the Data Flow

To confirm that your data is flowing, navigate back to Cribl Stream's Cribl HTTP Source. Run a **Live Data** capture on the Source.

You can also check the Monitoring page > **Data** submenu, to isolate the throughput on your Source.

12.1.5. Send Data To Loki And Prometheus

Prepare data for sending to Loki or Prometheus.

Loki and Prometheus, Grafana's systems for aggregating logs and for metrics-based monitoring, respectively, require specific setup to collect and store data coming from Cribl Edge.

Loki stores log contents unindexed, relying on labels and timestamps that are indexed to form event streams. The method is similar to how data would be stored in a bucket or index on other systems.

Similarly, Prometheus uses names, labels (dimensions), and numerical values.



For more straightforward setups where less customization is needed and you will ultimately use the data in Grafana Cloud, you can use the Grafana Cloud Destination.

Configure a Loki Pipeline

Loki stores logs based on labels. This is a great way to organize your stored data so that retrieval is easier when looking for specific log entries. By default, no labels are defined for the Loki Destination. Because Loki requires at least one label, Cribl will automatically add a default `label` source with the value `cribl_${__inputId}`.



See Loki Destination Processing settings for more information.

To ensure data is shaped correctly for Loki, you can configure a Pipeline with an Eval Function to add the labels to the `__labels` field.

To create a Pipeline called `loki-pipeline` in the Cribl Edge UI, follow these steps:

1. On the top bar, select **Products**, and then select **Edge**. Under **Fleets**, select a Fleet.
2. Select **More**, then **Pipelines**.
3. Select **Add Pipeline**.
4. In **ID**, enter `loki-pipeline` and provide an optional **Description**.
5. Confirm with **Save**.
6. Select **Add Function** and choose **Eval**.
7. In the **Evaluate Fields** section, add the configuration for your labels. For example, to prepare Kubernetes logs data, use:

Name	Value Expression
`__labels`	`{}`
`__labels.namespace`	`kube_namespace`
`__labels.pod`	`kube_pod`
`__labels.container`	`kube_container`

8. Select **Save**.

Next, in the Fleet's submenu, go to **Collect** to attach your Source (for example, a Kubernetes Logs Source) to Loki using the **loki-pipeline** Pipeline as connection.

Configure a Prometheus Pipeline

Cribl Edge offers a prebuilt `prometheus_metrics` Pipeline. You can modify it to change a few metadata fields and use it to connect to a Prometheus Destination.

1. On the top bar, select **Products**, and then select **Edge**. Under **Fleets**, select a Fleet.
2. Select **More**, then **Pipelines**, then **prometheus_metrics**.
3. Select the **Eval** Function and modify the fields in the **Evaluate Fields** section as needed. For example, to prepare Kubernetes metrics data, use:

Name	Value Expression
`job`	`job || 'cribl'`
`instance`	`__metadata.kube.node.metadata.name || host`

4. Select **Save**.

Next, in the Fleet's submenu, go to **Collect** to attach your Source (for example, a Kubernetes Metrics Source) to the Prometheus Destination, using the **prometheus_metrics** Pipeline as the connection.

12.2. Sources Overview

Each Cribl Edge **Source** is a configuration that enables Edge nodes to collect or receive observability data - logs, metrics, application data, and so on - in real time. Edge can receive continuous data input from Splunk, HTTP senders, Elastic Beats, Prometheus, TCP JSON, and many others. Sources can receive data from either IPv4 or IPv6 addresses.

Cribl Edge offers a configuration modal for each type of supported Source. However, you can add multiple instances of each Source type - with each configured to match the parameters of the corresponding sender. For example, you can have multiple File Monitors and multiple listeners for Syslog, Splunk, Elastic Beats, Prometheus, TCP JSON, and many others.

System and Internal Sources

Sources that generate data locally at the Edge Node, monitor resources, or move data among Edge Nodes and/or Stream Workers within your Cribl deployment.

- Cribl Internal
- Cribl HTTP
- Cribl TCP
- Datagen
- Exec
- File Monitor
- Journal Files
- Kubernetes Events
- Kubernetes Logs
- Kubernetes Metrics
- System Metrics
- System State
- Prometheus Edge Scraper
- Windows Event Logs
- Windows Metrics

Push Sources

Supported data Sources that **send** to Cribl Edge.

- Amazon Data Firehose
- Cloudflare (HTTP/S)
- Datadog Agent
- Elasticsearch API
- Grafana
- HTTP/S (Bulk API)
- Loki
- Metrics
- Model Driven Telemetry
- Raw HTTP/S
- OpenTelemetry (OTel)
- Prometheus Remote Write
- SNMP Trap
- Splunk HEC
- Splunk TCP
- Syslog
- TCP JSON
- TCP (Raw)
- UDP (Raw)
- Windows Event Forwarder
- Zscaler Cloud NSS

Comparison of Generic Push Sources

This table compares generic, protocol-level Push Sources. Refer to this table when you need to ingest raw data or custom formats over basic protocols like HTTP, TCP, or UDP.

Source	Key Differentiator	Best For...
HTTP/S (Bulk API)	**Protocol-aware:** Understands and responds using specific API semantics, including acknowledgments.	Receiving data from systems that require custom or generic HTTP/S event ingestion.

Source	Key Differentiator	Best For...
Raw HTTP/S	**Protocol-agnostic:** Accepts any HTTP/S payload without validating its structure.	Capturing any arbitrary or custom HTTP/S traffic where only the raw payload matters.
TCP JSON	**Content-aware:** Specifically designed to parse a stream of distinct JSON objects.	Ingesting a continuous stream of well-formed JSON objects over a reliable TCP connection.
TCP (Raw)	**Content-agnostic:** Treats the incoming data as an unstructured stream of bytes.	Ingesting binary data, custom application protocols, or any non-JSON data over TCP.
UDP (Raw)	**Stateless with best-effort delivery:** Offers lightweight, connectionless data ingestion.	High-volume, loss-tolerant data streams like syslog or SNMP traps.

Pull Sources

Cribl Edge does not contain any Sources categorized as **Pull**. The contents of the **Pull** tab will be blank.

Configuring and Managing Sources

For each Source *type*, you can create multiple definitions, depending on your requirements.

To configure Sources, from the top nav, select **Manage**, then select a **Fleet** to configure. Then, you have two options:

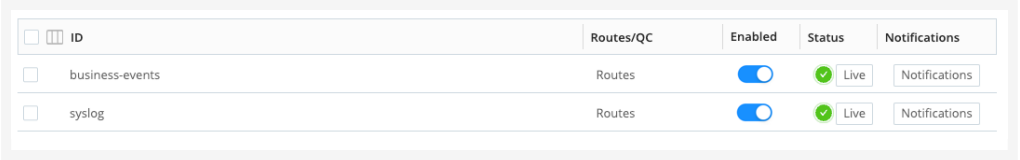
- To access the graphical QuickConnect UI, select **Collect**. Next, select either **Add New** or (if displayed), select **Existing**.
- To access the Routing UI, select **More** > **Sources**. On the resulting **Data Sources** page's tiles or left menu, select the desired type, then select **Add New**.

Capturing Source Data

To capture data from a single enabled Source, you can bypass the Preview pane, and instead capture directly from a **Manage Sources** page. Select the **Live** button beside the Source you want to capture.

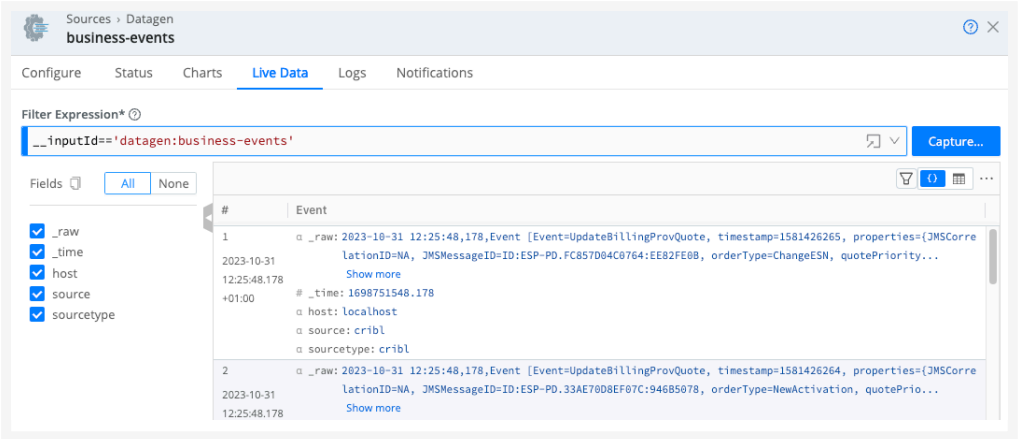


In order to capture live data, you must have Edge Nodes registered to the Fleet for which you're viewing events. You can view registered Edge Nodes from the **Status** tab in the Source.



Source > Live button

You can also start an immediate capture from within an enabled Source's configuration modal, by selecting the **Live Data** tab.



Source modal > Live Data tab

Monitoring Source Status

You can get a quick overview of Source health status by referring to their status icons.

Additionally, each Source's configuration modal offers two tabs for monitoring: Status and Charts.

Source Status Icons

Source status icons are available on the **Data** > **Destinations** page and for each individual Source in the list for a specific Source type.

The icons have the following meanings:

Icon	Meaning
	Healthy. Operating correctly.
	Warning. Experiencing issues. The Source is not functioning fully. Specific conditions will depend on the Source type.

Icon	Meaning
	Critical. Experiencing critical issues. Drill down to the Source's Status tab to find out the details.
	Disabled. The Source is configured, but not enabled.
	No health metrics available. This may mean that a Source is enabled, but has not been deployed yet.
	Inactive. When using GitOps, a Source appears `Inactive` if its **Environment** field (configured under **Advanced Settings**) does not match the currently active environment determined by the deployed Git branch. This ensures integrations only activate in their designated environments, preventing unintended data flow or misconfiguration.

Status Tab

The **Status** tab provides details about the Edge Nodes in the Fleet and their status. An icon next to each Edge Node uses color to clearly signal its health:

- : All systems go! Your Edge Node is operating normally.
- : Attention needed. There's a potential issue with this Edge Node.
- : Stop! This Edge Node has encountered a critical error.

The way you view Edge Node statuses depends on the size of your Fleet:

- **Fewer than 1000 Edge Nodes**: All Edge Nodes are conveniently displayed on the Status tab, along with their statuses. Use the **Status** checkboxes at the top to filter the list by health (healthy, warning, error). Select any Edge Node row to view specific details to help diagnose issues. The specific set of information provided depends on the Source type. Keep in mind that this data only reflects process `0` for each Edge Node.
- **More than 1000 Edge Nodes**: With a larger Fleet, you can select a specific Edge Node from the drop-down menu (showing up to 100 Nodes). You can search by `hostname` or `GUID` to find a specific Node. You can also use the **Status** checkboxes to filter which Edge Nodes appear in the drop-down list.



The content of the **Status** tab is loaded live when you open it and only displayed when all the data is ready. With a lot of busy Edge Nodes in a group, or nodes located far from the Leader, there may be a delay before you see any information.

The statistics presented are reset when the Edge Node restarts.

Charts Tab

The **Charts** tab presents a visual representation of Source activity using internal metrics. Unlike the status tab, this data is read almost instantly and does not reset when restarting an Edge Node.

The following data is available:

- Total events in
- Average throughput (events per second)
- Total bytes in
- Average throughput (bytes per second)

Configuring Time Ranges

At the top of the modal, a time range picker controls which time window the charts display:

- The picker offers the same presets as the main Monitoring time selector (for example, `5 min`, `15 min`, `1 hour`, `6 hours`, `1 day`).
- When you open the Source I/O metrics modal from **Monitoring**, the charts inherit the current Monitoring time range.
- When you open the modal from other areas of the UI (for example from **Data** > **Sources** or directly from a Source configuration modal), the Charts tab defaults to `10 minutes`.
- Changing the time range in the modal only affects that modal and is not persistent. It does not change the global time range on the originating page (for example, the **Monitoring** overview or the page-level time picker).

Use this picker to zoom in on short spikes or expand to longer windows when you're troubleshooting Source performance or throughput issues.

Preconfigured Sources

To accelerate your setup, Cribl Edge ships with several common Sources configured but not switched on. Open, clone (if desired), modify, and enable any of these preconfigured Sources to get started quickly.

> On a Cribl.Cloud deployment, **do not delete** any preconfigured Sources. If you don't plan to use them, keep them disabled.

These default Sources are managed by Cribl Edge and are re-created from the default configuration when you create or clone Fleets, upgrade, or apply certain configuration bundles. As a result, they cannot be permanently

deleted. Removing them in the UI only affects the current configuration copy, and they will reappear the next time defaults are re-applied.

- **System Metrics** - Basic Level
- **System State** > **in_system_state**
- **File Monitor** > **in_file_auto** - Auto Discovery Mode
- **File Monitor** > **in_file_varlog** - Manual Discovery Mode
- **Kubernetes Events** > **in_kube_events**
- **Kubernetes Logs** > **in_kube_logs**
- **Kubernetes Metrics** > **in_kube_metrics**
- **Cribl Internal** > **CriblLogs** - Internal
- **Cribl Internal** > **CriblMetrics** - Internal
- **Windows Metrics** > **in_windows_metrics**
- **Windows Event Logs** > **in_win_event_logs**
- **Journal Files** > **in_journal_local**

Single Source Support

The following Source types are limited to preconfigured Sources:

- Cribl Internal
- Kubernetes Events

12.2.1. System

You can use the following Sources to receive data from internal senders.

Source	Description
Exec	Execute a command periodically, and collect its stdout output
File Monitor	Collect log files and generate events from the file content
Journal Files	Collects data from systemd's journald service
Kubernetes Events	Generate events for a Kubernetes cluster
Kubernetes Logs	Collects logs from containers on a Kubernetes node
Kubernetes Metrics	Generate events for Kubernetes clusters, nodes, Pods, and containers
System Metrics	Collect metrics from a host, and populate dashboards
System State	Collects the system's current state on a configurable schedule, relaying corresponding events to downstream systems.

12.2.1.1. Exec Source

The Exec Source enables you to periodically execute a command and collect its `stdout` output. This is typically used in cases when Cribl Edge cannot accomplish collection with native Collectors or other Sources.

Type: **System** | TLS Support: **N/A** | Event Breaker Support: **Yes**

This Source is available in Cribl.Cloud on customer-managed hybrid Workers, but not on Cribl-managed Workers in Cribl.Cloud.

This Source allows you to run **almost anything** on the host system. Make sure you understand the security and other impacts of commands you plan to execute, before proceeding. To disable it on your Edge Nodes, see Disable the Exec Source below.

Especially for monitoring or polling, you'll need to receive the command or script's output periodically. For this reason, the Exec Source lets you specify when the command or script should run, by time interval or by cron-style schedule.

Here are a few examples of what you can run from an Exec Source:

- Database queries.
- Custom commands to poll databases or apps.
- `ping` to check latency.
- `ps -ax` to get a list of running processes.
- `SNMP GET` requests to query an SNMP agent about network entities.
- `netstat -natp` to get lists of sockets and TCP ports, and what they're doing.
- `mtr -c 1 --report --json 8.8.8.8` to run a traceroute on a location (here, `8.8.8.8`), and packaging up a report in JSON format.

Configure an Exec Source

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Source** and select the Source you want from the list, choosing either **Select Existing** or

Add New.

- To configure via the Routes, select **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). Select the Source you want. Next, select **Add Source**.

2. In the **New Source** modal, configure the following under **General Settings**:
   - **Enabled**: Defaults to toggled on.
   - **Input ID**: Enter a unique name to identify this Exec Source definition. If you clone this Source, Cribl Edge will add `-CLONE` to the original **Input ID**.
   - **Command**: Command to execute; supports Bourne shell syntax.
   - **Schedule type**: Use the buttons to select either `Interval` or `Cron`. For details, see Schedule Type below.
3. Next, you can configure the following **Optional Settings**:
   - **Script**: Optional multiline script to send to the defined command. The content of this field will be piped into the `stdin` of the command. The `stdin` stream is closed after adding the script. See Run Custom Scripts for more information.
   - **Retry limit**: Maximum number of retry attempts in the event that the command fails.
   - **Tags**: Optionally, add tags that you can use to filter and group Sources in Cribl Edge's UI. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
4. Optionally, you can adjust the Processing and Advanced settings, or Connected Destinations outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

Schedule Type

Customize the behavior in the corresponding field below the button.

- **Interval**: Specify how often, in seconds, the command should run. Defaults to `60`.
- **Schedule**: Enter a cron expression. (You enter the cron expression in UTC time, but the resulting **Estimated Schedule** displays in local time.)

Processing Settings Event Breakers

Event Breaker rulesets: A list of event breaking rulesets that will be applied to the input data stream before the data is sent through the Routes. Defaults to `System Default Rule`.

Event Breaker buffer timeout: How long (in milliseconds) the Event Breaker will wait for new data to be sent to a specific channel, before flushing out the data stream, as-is, to the Routes. Minimum `10` ms, default `10000` (10 sec), maximum `43200000` (12 hours).

Fields

In this section, you can define new fields or modify existing ones using JavaScript expressions, similar to the Eval function.

- The **Field Name** can either be a new field (unique within the event) or an existing field name to modify its value.
- The **Value** is a JavaScript expression (enclosed in quotes or backticks) to compute the field's value (can be a constant). Select this field's advanced mode icon (far right) if you'd like to open a modal where you can work with sample data and iterate on results.

This flexibility means you can:

- Add new fields to enrich the event.
- Modify existing fields by overwriting their values.
- Compute logic or transformations using JavaScript expressions.

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline or Pack to process data from this input before the data is sent through the Routes.

Advanced Settings

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

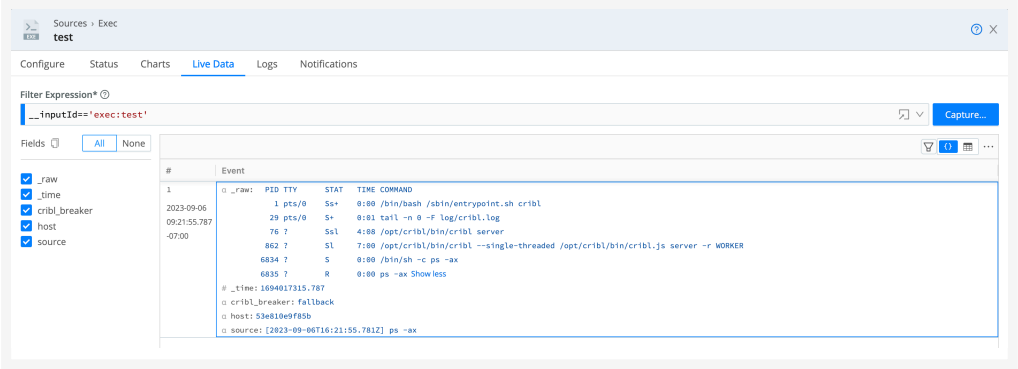
Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

Internal Fields

Cribl Edge uses internal fields to assist in forwarding data to a Destination.

You can find the following event fields on the **Live Data** tab of the selected Exec Source, including the hostname and source of the event:

- `_raw`
- `_time`
- `cribl_breaker`
- `host`
- `source`



Event fields

Troubleshooting

If a command isn't running as expected, the **Logs** tab can help you identify issues by displaying the command executed, its elapsed time, and its exit code.



Event fields

Run Custom Scripts

You can use the Exec Source to run custom multiline scripts, for example in Bash or Python. To do so, enter the command in the **Command** field and the script contents in the **Script** field.

For example, to run a Bash script, enter:

Command: `bash -`

Script:

```
#!/usr/bin/env bash
USER_NAME=$(whoami)
DATE=$(date)
echo "Hi $USER_NAME, today is $DATE"
```

To run a Python script, enter:

Command: `python3 -`

Script:

```
#!/usr/bin/env python3
import getpass
from datetime import datetime
user_name = getpass.getuser()
date_now = datetime.now()
print(f"Hi {user_name}, today is {date_now}")
```

Execute Windows PowerShell Commands

On a Windows Edge Node, the Exec Source command is executed from a Windows command prompt, not from a PowerShell prompt. However, you can still run PowerShell commands in it with some modification.



The execution of PowerShell is resource-intensive and should be used with caution. On limited resource systems (e.g., domain controllers) it is best to run Exec with PowerShell carefully as the Source uses the host's CPU.

To run a PowerShell command in the Exec Source, you need to prepend the **Command** field with `powershell`, for example:

PowerShell Prompt Command	Cribl Exec Source configuration
`Get-Process`	`powershell Get-Process`
`# cd c:\mydir` `# .\myps-script.ps1`	`powershell c:\mydir\myps-script.ps1`

You can handle a PowerShell command with parameters by placing the command and the parameters in double quotes (`""`), for example:

PowerShell Prompt Command	Cribl Exec Source configuration
`Get-CimInstance win32_process | select ProcessId, Name, commandline, HandleCount, WorkingSetSize, VirtualSize`	`powershell [-command] "Get-CimInstance win32_process | select ProcessId, Name, commandline, HandleCount, WorkingSetSize, VirtualSize"`

(Include the `-command` parameter if your PowerShell version requires it.)

Disable the Exec Source on a Node

To disable the Exec Source on a specific Edge Node, set the `CRIBL_NOEXEC` environment variable. This will override any configuration pushed from the Leader Node, ensuring the Exec Source remains disabled on that particular node.

The `CRIBL_NOEXEC` environment variable must be set outside of the Cribl product to disable the Exec Source. This variable cannot be modified within the product's interface.

How to Disable the Exec Source on Different Platforms

To disable the Exec Source, follow the platform-specific instructions below. Once you've set the `CRIBL_NOEXEC` environment variable, the Exec Source will be disabled. To re-enable it, remove or unset the variable.

Disable on Linux Platform

To disable the Exec Source on a Linux system running as a Systemd service, edit the appropriate service definition file:

- Cribl Edge: `/etc/systemd/system/(cribl|cribl-edge).service`
- Cribl Stream: `/etc/systemd/system/cribl.service`

To ensure the `CRIBL_NOEXEC` environment variable is set when the Cribl Edge service starts, add `CRIBL_NOEXEC=1` to the service's configuration. If the service is currently running, a restart is required for the

change to take effect.

Disable on Docker

To disable the Exec Source on Docker, modify the `docker run` command (or equivalent) to include the `-e CRIBL_NOEXEC=1` flag.

Disable on Kubernetes/Helm

To disable the Exec Source on Kubernetes, add the `--set env.CRIBL_NOEXEC=1` flag to your `Helm install` command or modify the `values.yml` file to include this setting.

Disable on Windows (For Cribl Edge)

To add the `CRIBL_NOEXEC` environment variable to Cribl Edge on Windows, use the command prompt to execute the following command:

```
setx CRIBL_NOEXEC "1" /M
```

This command sets the environment variable for the current user account. The Cribl Edge service runs under `LocalSystem` user account by default.

12.2.1.2. File Monitor Source

The File Monitor Source generates events from text files, compressed, archived, and (some) binary log files, based on lines and records extracted from the content.

Type: **System** | TLS Support: **N/A** | Event Breaker Support: **Yes**

This Source is available in Cribl.Cloud on customer-managed hybrid Workers, but not on Cribl-managed Workers in Cribl.Cloud.

As of v.4.2.x, this Source can process the following file formats:

- **Compressed/archived**: `zip`, `gzip`, `zstd`, and `tar`. Cribl Edge only supports DEFLATE compression for `zip` files.
- **UTF-8**: `text`.
- **Non-identifiable binary**: Includes all file formats that aren't in the Cribl Edge-supported compressed, archived, and text categories.

Binary files are broken into base64-encoded chunks and streamed bypassing Event Breakers. Text files are processed normally through Event Breakers. (Before v.4.2.x, the File Monitor Source could handle only text files.) Also as of v.4.2.x, the File Monitor Source no longer excludes `*.gz` files by default.

File Monitoring and One-Time Ingestion

Cribl Edge offers two options for ingesting file data:

- **File Monitor Source**: This is ideal for ongoing log file monitoring. By default it will prioritize processing the latest data in newly discovered files upon startup ("Collect from end"). Subsequently, it automatically monitors and ingests new content as it's added.

To automatically discover and ingest new log files, Cribl Edge requires privileged access for file scanning and reading. For alternative approaches, see Running Edge as an Unprivileged User.

- **One-time File Ingest**: This feature complements the File Monitor by allowing a one-time ingest of any file. It's useful for quickly processing specific files or initial data loads. If **Explore** is enabled, as it might be restricted in some environments for security reasons.

Discovering and Filtering Files to Monitor

To produce its initial list of files to monitor, the File Monitor Source runs a discovery procedure at a configurable **Polling interval**. The Source then applies a **Filename allowlist** to filter the initial list down into its final form. Then, for each file on the list, the Source compares current state with previously-stored state. This comparison determines whether the File Monitor Source will actually watch a given file for a given polling interval, or just ignore the file.

In the simplest case, the Source discovers a file for which it has no stored state. This means that the file has just been created, and needs to be monitored. See the Examples for other possibilities, along with a description of the **Status** tab, which displays state information for all files being monitored.

How does the File Monitor Source discover files in the first place? You have the choice of two **Discovery Modes**: **Auto** or **Manual**.

- In **Auto** mode, the Source automatically discovers files that running processes have open for writing. Auto mode collects logs from the discovered active files that match the allowlist.
  Auto mode is useful to detect which files are being written to. For example, if you enter an allowlist of `*log`, Auto mode will find all the active logs within that directory, no matter what's writing to it, and it will exclude any rotated logs.



Cribl Edge currently supports Auto mode only when running on Linux, not on Windows or macOS. Windows and macOS deployments require using the Manual option. For a Windows-specific configuration, see Collect Logs with File Monitor.

- In **Manual** mode, the Source discovers files based on the specified directory and depth.
  Manual mode is often used to collect logs in a folder where the process that creates them rotates them when they get large. For example, if you specify a path of `/var/log` and an allowlist of `*/messages*`, Manual mode will capture the current `/var/log/messages`, as well as older logs from `/var/log/messages*`.
  Manual is the default discovery mode for new File Monitor Sources.



If you are using a tool like `rsync` to transfer files in chunks, the File Monitor Source might start collecting files before the transfer is complete. To prevent this, configure your tool to transfer files serially. (For example, use: `rsync --append`.)

Configure a File Monitor Source

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Source** and select the Source you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). Select the Source you want. Next, select **Add Source**.
2. In the **New Source** modal, configure the following under **General Settings**:
    - **Enabled**: Toggle on to enable the Source.
    - **Input ID**: Enter a unique name to identify this File Monitor Source definition. If you clone this Source, Cribl Edge will add `-CLONE` to the original **Input ID**.
    - **Description**: Optionally, enter a description.
    - **Discovery Mode**: Select either **Auto** or **Manual**. See Discovering and Filtering Files to Monitor for details on how each mode works.
        - **Auto**: Tells the Source to automatically discover files that running processes have open for writing.
        - **Manual**: Tells the Source to discover files within a single directory that you specify as the **Search path**, down to the **Max depth**.
            - **Search path**: Enter an absolute path (no wildcards). Defaults to the `/var/log` directory.
            - **Max depth**: Controls how many levels deep the Source will search within the **Search path**. If you leave the **Max depth** field empty, the Source will search subdirectories, and their subdirectories, and so on, without limit. If you specify `0`, the Source will discover only the top-level files within the **Search path**. For `1`, the Source will discover files one level down.
            - **Suppress errors when search path does not exist**: Toggle on to hide `Path does not exist` errors when a directory doesn't exist. When enabled, missing directories will not affect the File Monitor Source health status. This is useful when you know the search path contains nonexistent directories for some hosts, and you don't want to see the error messages in the File Monitor status and logs.
            - **Delete files**: Whether to delete the source file after it has been idle for the specified **Idle timeout** period. This toggle is under Advanced Settings.
3. Next, you can configure the following **Optional Settings**:
    - **Polling interval**: Interval of time in seconds to run the discovery procedure. Defaults to `10` seconds.

- **Filename allowlist**: Controls which files the Source discovers and monitors. Defaults to `*/log/*` and `*log`. You can use wildcards (`*`) to match any sequence of characters and the exclamation mark (`!`) for negation. These patterns are matched against the full path of the file. For example:
  - Wildcard (`*`): `*DhcpSrvLog-*.log` collects file `/var/log/DhcpSrvLog-05-22-2025.log`.
  - Negation `!`: `!*cribl*access.log` prevents the Source from discovering Cribl Edge's own log files.



If you're using **Manual** discovery mode, always think through how your **Search path** and **Filename allowlist** settings interact. If you're not careful, you can inadvertently create overlapping monitor groups that deliver duplicates of some files. See Using Allowlists Effectively for details.

- **Apply filename allowlist internal to archive files**: Toggle on if you want the filename allowlist to apply to files inside archives such as `tar` or `zip`.
- **Minimum age duration**: Optionally, specify a minimum age of files to monitor. This Source will only start ingesting a file once its modification time exceeds the configured duration. Enter a duration in a format like 30s, 15m, 1h. Defaults to an empty field, which applies no age filters.
- **Maximum age duration**: Optionally, specify a maximum age of files to monitor. This Source will filter events with timestamps newer than the configured duration. (Where files don't have a parsable timestamp, it will set their events' timestamp to `Now`.) Enter a duration in a format like `30s`, `4h`, `3d`, or `1w`. Defaults to an empty field, which applies no age filters.
- **Check file modification times**: Toggle on if you want this Source to skip files with modification times older than the configured **Maximum age duration**. When you set an **Maximum age duration** threshold, the Source will open every file and read through all its contents to seek a timestamp. For best performance, also toggle **Check file modification times** on skip older files.
- **Collect from end**: When toggled on (default) this Source prioritizes processing the latest data by jumping to the end of newly discovered files on Cribl Edge's startup. This allows focusing on new content in existing log files. However, this behavior only applies during the first encounter with a file. For existing files after the initial scan and all subsequent discoveries, Cribl Edge will switch to reading from the beginning to capture ongoing updates.
- **Enable binary files**: Toggle on if you want this Source to report the file as binary and stream it in Base64-encoded chunks. By default, the Source ignores binary (non-text) files such as JPEG images, MP3 audio files, or some binary data files.
- **Force text format**: Toggle on if you want this Source to display ingested data as text in the event (when the data isn't in a compressed or archived file). This option is helpful if you have files that contain binary mixed with text where the data should be streamed as all text (for example, AuditD logs can contain this type of data).
- **Tags**: Optionally, add tags that you can use to filter and group Sources in Cribl Edge's UI. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

4. Optionally, you can adjust the Processing and Advanced settings, or Connected Destinations outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

Processing Settings Event Breakers

Event Breaker rulesets: A list of event breaking rulesets that will be applied to the input data stream before the data is sent through the Routes. Defaults to `System Default Rule`.

Event Breaker buffer timeout: How long (in milliseconds) the Event Breaker will wait for new data to be sent to a specific channel, before flushing out the data stream, as-is, to the Routes. Minimum `10` ms, default `10000` (10 sec.), maximum `43200000` (12 hours).

Fields

In this section, you can define new fields or modify existing ones using JavaScript expressions, similar to the Eval function.

- The **Field Name** can either be a new field (unique within the event) or an existing field name to modify its value.
- The **Value** is a JavaScript expression (enclosed in quotes or backticks) to compute the field's value (can be a constant). Select this field's advanced mode icon (far right) if you'd like to open a modal where you can work with sample data and iterate on results.

This flexibility means you can:

- Add new fields to enrich the event.
- Modify existing fields by overwriting their values.
- Compute logic or transformations using JavaScript expressions.

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline or Pack to process data from this input before the data is sent through the Routes.

Advanced Settings

Idle timeout: Time, in seconds, before an idle file is closed. Defaults to `300` sec. (5 minutes).

Hash length: How many file header bytes to use in a hash for identifying a file uniquely. Defaults to `256` bytes. For details, see Configuring Hash Lengths.

Delete files: Default is toggled off. When toggled on: Cribl Edge will delete the source file after it has been idle for the specified **Idle timeout** period. This setting is only available in `Manual` mode. In the confirmation modal, select **Enable** to proceed. Files that fail to delete will be reported as errors on the **Status** tab for up to one hour.

Files might not be deleted if:

- The Cribl Service doesn't have proper permissions to access and delete the files.
- They are currently in use by other applications on Windows.
- They are being moved or renamed while Cribl Edge is collecting them.
- Their filenames are not on the allowlist.

To make sure files are deleted properly:

- Verify and grant necessary permissions to the Cribl Service. For details, see Configure Cribl Stream for Non-Root User and Running Edge as an Unprivileged User.
- Consider access patterns and how files are being used.
- Adjust system settings if necessary (such as, file sharing, antivirus exclusions).

Salt file hash: Default is toggled off. Toggle on to salt the hash with the source file name. Use this option for files with the same header (for example, CSV files) to ensure that each has a different hash and can be ingested correctly. A file will get a new hash and be re-ingested if you move or rename it. Additionally, toggling this setting on after the Source has started operating will cause all files to be re-ingested as well.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Discovery Optimizations

The **Discovery Optimizations** section includes settings for optimizing the file discovery process.

These settings apply only in specific situations and can have an adverse effect when used incorrectly. Consult with Cribl Support to determine whether these settings are appropriate for your needs.

Optimize for leaf directories: Default is toggled off. Toggle on to reduce load on the filesystem for search paths with files in leaf directories. This will cause file discovery to skip rescanning unchanged directories (determined based on modification time), using an exponential backoff strategy. This might delay detection of new data.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.



This Source defaults to QuickConnect.

Internal Fields

Cribl Edge uses a set of internal fields to assist in handling of data. These "meta" fields are not part of an event, but they are accessible, and Functions can use them to make processing decisions.

For the File Monitor Source, you can use internal fields to enrich events with container and host metadata. The following internal fields are available:

- `__baseFilename`
- `__source` / `source`
- `__raw`
- `__inputId`
- `container_id`
- `container_path`
- `host_path`
- `__unresolved_source` (This internal field represents the unresolved path for the symbolic link configured in the File Monitor Source.)

Monitoring Renamed Files' State

The File Monitor Source uses a simple scheme to find "backlog" files corresponding to a matching file: Tell it to tail `foo.log`, and it will look for `foo.log.[0-9]` and scrape those, too.

This Source keeps hashes that correspond to the start point within the file, and to the last-read point. This way, if Cribl Edge is stopped, files are rotated, and Cribl Edge is restarted, the File Monitor Source can find the resume point in those backlog files.

Similarly, if you rename the file **within the same backlog scheme** (for example, `foo.log` to `foo.log.0`), the File Monitor Source can resume at the right place. However, if you renamed `foo.log` to `bar.log` (a deviation from the expected naming scheme), this Source could not find its resume point.

File Tracking with Hashes

The File Monitor Source doesn't track state using filenames. Instead, it uses hashes of the head and tail of the file to identify it later so it can resume where it left off.

To make sure hashes are generated correctly to ingest the files you want, follow these better practices.

Configure Hash Lengths

For files with identical first lines, configure the hash length to be longer than the first line to ensure that the header includes something unique. It is common for a batch of `CSV` files or `IIS` logs to have identical first lines listing columns that appear in subsequent lines.

If you don't adjust the hash length, all files with headers larger than `256 bytes` will appear to File Monitor as one file. As long as the hash length is greater than the header length, and the subsequent lines include something unique, the File Monitor Source will identify them as different files.

Salt the Hashes

If you want to ingest files with identical first lines (such as CSV files), and configuring hash length does not yield satisfying results, you can ensure different hashes for the files by toggling the **Salt file hash** setting. See Salt file hash for more details.

Avoid Race Conditions

The hashing logic isn't instantaneous, so it's possible for files with the same header and the same hash to be both collected. For example, quickly copying the same example file to separate `eg1.log` and `eg2.log` files in the Source's path will usually result in both files being collected even though they share the same hash. However, copying it to `eg3.log` shortly (not immediately) after will result in that third file being ignored because its hash matches an existing entry in the state-store.

Don't Ingest Multiple Copies of the Same File

Creating copies of the same file in the Source's path will not result in the file being ingested multiple times. The head-hashing logic used by the Source to determine what it's seen before and how far it's collected already prevents you from successfully testing multiple copies of the same.

Don't Point Multiple File Monitor Sources at the Same File

Multiple File Monitor Sources set to collect the same file will result in duplicate events. Sources run independently and don't coordinate with each other.

Using Allowlists Effectively

Done properly, file monitoring collects exactly the files you need to see, without duplication. You want to avoid inadvertently creating overlapping monitor groups that deliver duplicates, because this can "run the meter" twice for affected files, potentially causing issues with data ingestion volume limits.

The key is knowing how to craft allowlists.

Crafting Allowlists: Principles

To get the results you want from file monitoring, always apply the following principles.

Write allowlists to explicitly include the full path of files you want to collect.

The search path itself is **not** excluded from the match. For example, if you want to collect the file `/var/log/messages` when you've set your search path to `/var/log`, your filename allowlist needs to be either `/var/log/messages`, `/var/*/messages`, or `*/log/messages`.

Exclusions (`!`) also need to address the full file path. For example, if you want to collect everything from `/var/log/*` **except for** `/var/log/apache/*`, your allowlist must include one of the following:

- `!/var/log/apache/*`
- `!*/apache/*`
- `!*/log/apache/*`

Wildcards (`*`) match **any** part of the file path:

- `*log` matches `/var/foo/bar/myfile.log`.
- `*.log` matches anything ending in `.log`.
- `/var/log/*` matches anything in `/var/log`, subject to the **Max Depth** setting.

Put explicit matches (and exclusions) at the beginning of the allowlist, ahead of wildcard matches.

For example, an allowlist of `/var/log/*` followed by `!/var/log/apache/*` will fail to exclude the file `/var/log/apache/foo` because that file matches the first allowlist entry.

In allowlists, order matters!

Be as specific as possible, whenever possible, to prevent accidental matches.

Each of the following examples shows how to avoid matching unwanted files within the directories that you're monitoring.

To collect `/var/log/boot.log` (and older versions of `boot.log.*`) as well as `/var/log/messages`:

Search path	Filename allowlist
`/var/log`	`/var/log/boot.log`, `/var/log/messages`

To collect everything in `/var/log` while excluding `/var/log/apache/*`:

Search path	Filename allowlist
`/var/log`	`!/var/log/apache/*`, `/var/log/*`

Examples of Allowlists

Below are typical scenarios where good allowlist technique will save you from duplication headaches.

Example: Excluding the System's Own Logs

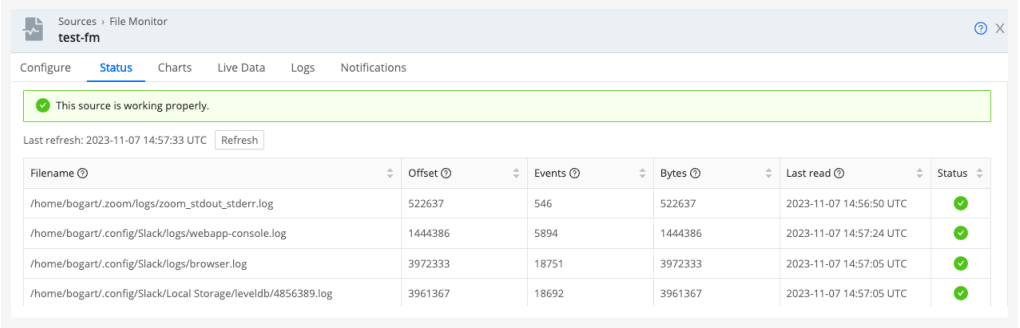
Recall that you'll often create a set of File Monitor Sources, each of which monitors one file. This is **not** the right approach in an example like this one, where **Discovery Mode** is set to **Auto**. Using **Auto** mode in a file monitor along with literally any other file monitor can cause unintended, duplicate collection. Use **Auto** mode carefully!

Here's the example:

Suppose you added a File Monitor Source on a Linux machine, set your **Discovery Mode** to **Auto**, and specified your home directory as the **Search path** (for example, `/home/bogart/`).

You do not want to monitor Cribl Edge's own log files, or any log files generated by Chrome, the web browser you're using. To exclude them, you add `!*cribl*access.log` and `!*chrome*` to your **Allowlist**. You **do** want to monitor any other log files that the Source can discover, so you also add `*log`.

After awhile, the Status tab looks like this:



The Status tab

Example: Fixing Overlapping Monitor Groups

Here's an example of the kind of overlapping monitor groups you should avoid:

Search path	Max depth	Filename allowlist
`/var`	(none)	`*/log/*`
`/var/log/apache`	`0`	`*`

The above configuration would collect all the files in `/var/log/apache/` twice. A better solution would be the following:

Search path	Max depth	Filename allowlist
`/var/log`	(none)	`!*/apache/*`
`/var/log/apache`	`0`	`*`

Troubleshooting

Common mistakes to avoid when exploring the File Monitor Source include the following.

Don't Manually Edit Ingested Files

Do not test the File Monitor Source by editing and saving files using a text editor (such as vi or Notepad). Under the hood, these editors operate on a temporary file that they move into place when saved. As a result, the Source sees files get replaced and moved rather than appended to. This confuses the Source's logic that assumes logs are appended to the existing file.

Instead, to test the Fole Monitor Source, run the command below to append logs to files: `echo "{ \"time\":\"$(date -Is -u)\", \"message\":\"...\" }" >> eg.log`

You end up with lines like this in the logs:

```
{ "time":"2023-09-19T20:08:42+00:00", "message":"..." }
```

Test File Ingestion with Event Breakers

To make sure that incoming files are split into individual events as expected, test sample file content with Event Breakers. When you configure the Event Breaker Ruleset, configure the timestamp correctly and verify that it works. The Event Breaker will default to "Now" if it can't find a timestamp in the raw log, and it saves the cursor state at that point.

If the Event Breaker encounters another log later in the file with a timestamp before “Now”, it will ignore it. For

12.2.1.3. Journal Files Source

The Journal Files Source collects data from systemd's centralized logging, which is called the journald service. This service is a centralized location for all messages logged by different components in a systemd-enabled Linux system. This includes messages from kernel, boot, syslog, or other services. This Source is currently configured to configured to collect data from the `user` journal, as `system` is privileged. This Source is disabled by default.

Type: **System** | TLS Support: **N/A** | Event Breaker Support: **No**

The Journal Files Source is available in Cribl.Cloud on customer-managed hybrid Workers, but not on Cribl-managed Workers in Cribl.Cloud.

Discovering and Filtering Journal Files to Monitor

To determine all the names of all journals to read, the Journal Files Source runs a discovery procedure at a configurable **Polling interval**. The Source then applies an **Allowed list** to filter the initial list down into its final form. Then, for each journal on the list, the Source compares current state with previously stored state. This comparison determines whether the Journal Files Source will actually stream a given journal for a given polling interval. The Source then runs the files through the **Filter Rules** which allows you to set additional criteria.

Cribl Edge ships with a single, disabled instance of the Journal Files Source with a default configuration. The search path is configured to point to `$CRIBL_EDGE_FS_ROOT/var/log/journal/$MACHINE_ID` and defaults to streaming the primary `system.journal` it finds there. `$MACHINE_ID` is an environment variable that you can insert into the Source's search path. If you use the environment variable but don't assign a value, the Source will look up the local host `machine id` from `/etc/machine-id` and use that value.

Currently, this Source does not support all of the compression options that the journald service supports. The parser this Source uses to read the files can handle `LZ4` and `ZSTD` compression. When the parser encounters compressed fields it can't unpack, an error message emits in the logs. Below is an example of the error message:

```
Error reading journald event, dropping offset=31678152 evt="Mar 02 21:29:54
goats-xps15 multipassd: undefined"
```

Configure a Journal Files Source

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Source** and select the Source you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). Select the Source you want. Next, select **Add Source**.
2. In the **New Source** modal, configure the following under **General Settings**:
    - **Enabled**: Toggle on to enable the Source.
    - **Input ID**: Enter a unique name to identify this Source configuration. If you clone this Source, Cribl Edge will add `-CLONE` to the original **Input ID**.
    - **Description**: Optionally, enter a description.
    - **Search path**: Specify the directory to search journals on. Accepts environment variables. The default value for this field varies depending on the product:
    - In Cribl Edge, it defaults to `$CRIBL_EDGE_FS_ROOT/var/log/journal/$MACHINE_ID`.
    - In Cribl Stream, it defaults to `/var/log/journal/$MACHINE_ID`.
    - **Journal allowlist**: The full path of each discovered journal is matched against this wildcard list. Defaults to `system`.
3. Next, you can configure the following **Optional Settings**:
    - **Polling interval**: How often, in seconds, to collect metrics. If not specified, defaults to `10` seconds.
    - **Filter Rules**: Optionally, specify which journal objects to allow. Events are generated if all the rules' expressions evaluate to true, or if no rules are specified.
    - **Filter expression**: JavaScript expression to filter Journal objects. Returns `true` to include it. For details, see Filter Expressions below.
    - **Description**: Optional description of the rule.
    - **Current boot only**: Skip events that are not part of the current boot session.
    - **Age duration limit**: Optionally, specify a maximum age of journal objects to stream. This Source will filter events with timestamps newer than the configured duration. Enter a duration in a format like `30s`, `4h`, `3d`, or `1w`. Defaults to an empty field, which applies no age filters.
    - **Tags**: Optionally, add tags that you can use to filter and group Sources in Cribl Edge's UI. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

4. Optionally, you can adjust the Processing and Advanced settings, or Connected Destinations outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

Filter Expressions

The default **Filter expression** is `severity <= 4` with a **Description** of `Allow events having 'emergency', 'alert', 'critical', 'error', or 'warning' priority.`

`The screenshot below shows other examples you can use:`



Filter expressions

Processing Settings Fields

In this section, you can define new fields or modify existing ones using JavaScript expressions, similar to the Eval function.

- The **Field Name** can either be a new field (unique within the event) or an existing field name to modify its value.
- The **Value** is a JavaScript expression (enclosed in quotes or backticks) to compute the field's value (can be a constant). Select this field's advanced mode icon (far right) if you'd like to open a modal where you can work with sample data and iterate on results.

This flexibility means you can:

- Add new fields to enrich the event.
- Modify existing fields by overwriting their values.
- Compute logic or transformations using JavaScript expressions.

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline or Pack to process data from this input before the data is sent through the Routes.

Advanced Settings {advanced-tab}

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.



In Cribl Edge, this Source defaults to QuickConnect. While, in Cribl Stream, it defaults to Send to Routes.

12.2.1.4. Kubernetes Events Source

The Kubernetes Events Source relies on the Kubernetes API watch capabilities to collect cluster-level events. These events are generated automatically in response to state changes or errors with nodes, pods, and containers. The Kubernetes API server acts as a notification server for the events we are watching in a given Kubernetes cluster. This Source acts in a listener/subscriber capacity to collect events for all namespaces from the cluster API. For a sample of the events captured, see Live Data.

To collect cluster-level events, the Kubernetes Events Source needs watch access to the Kubernetes API. For details, see RBAC for Kubernetes Events Source.



Type: **System and Internal** | TLS Support: **N/A** | Event Breaker Support: **No**

You can't configure custom Kubernetes Events Sources.

Discover and Filter Kubernetes Events

The Kubernetes Events Source connects to the Kubernetes API to set up a watch, and to request a starting point from the latest event at 5-minute polling intervals. The Source then runs the lists through the **Filter Rules** to determine what to report on.

This Source emits events in a way that might include some redundancies in the properties extracted. For details on the output, plus guidelines on what fields to drop, see Event Structure Details.

If no rules are configured, or if all of the rules evaluate to `true`, this Source generates events for the object. Conversely, if any of the rules evaluates to `false`, this Source skips the object, and does not generate any events for it.

Cluster/Daemonset Best Practices

For the Kubernetes Events Source to work as designed, Cribl recommends deploying Cribl Edge as a Daemonset. For details, see Deploying via Kubernetes.

This Source can generate redundant events when it is running inside the cluster, and can't determine which DaemonSet it's in or which Node it's on. This happens when you run Cribl Edge inside the cluster, but in a way that it can't detect the local Pod/node. In these cases, Edge will emit an error when it initializes this Source.

Configure Cribl Edge to Collect Kubernetes Events

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Collect**. Select **Add Source** and select the Source you want from the list, then choose **Select Existing**.
    - To configure via the Routes, select **More** > **Sources**. Select the Source you want.
2. Select the default `in_kube_events` to open the configuration modal.
3. Configure the following under **General Settings**:
    - **Input ID**: This is prefilled with the default value `in_kube_events`, which cannot be changed via the UI, due to the single-Source restrictions above.
    - **Description**: Optionally, enter a description.
4. Next, you can configure the following **Optional Settings**:
    - **Filter Rules**: Optionally, specify the Kubernetes objects which this Source should parse to generate events. If you specify no restrictive rules here, the Source will emit all events.
    - **Filter expression**: JavaScript expression to filter Event objects. For details, see Filter Expression below.
    - **Description**: Optional description of the rule.
    - **Tags**: Optionally, add tags that you can use to filter and group Sources in Cribl Edge's UI. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
5. Optionally, configure any Processing and Advanced settings, or Connected Destinations outlined in the sections below.
6. Select **Save**, then **Commit & Deploy**.

Filter Expression

Filters are based on the Event Objects, and evaluated from top to bottom. The first expression evaluating to `false` excludes the Pod from collection.

Filter expressions examples include:

- Ignores events in the `kube-*` namespaces - `!object.metadata.namespace.startWith('kube-')`
- Ignore all events from DaemonSet - `object.regarding.kind != 'DaemonSet'`
- Collects events matching a specific object name - `object.regarding.name == 'object name'`

Processing Settings Fields

In this section, you can define new fields or modify existing ones using JavaScript expressions, similar to the Eval function.

- The **Field Name** can either be a new field (unique within the event) or an existing field name to modify its value.
- The **Value** is a JavaScript expression (enclosed in quotes or backticks) to compute the field's value (can be a constant). Select this field's advanced mode icon (far right) if you'd like to open a modal where you can work with sample data and iterate on results.

This flexibility means you can:

- Add new fields to enrich the event.
- Modify existing fields by overwriting their values.
- Compute logic or transformations using JavaScript expressions.

Advanced Settings

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Connected Destinations

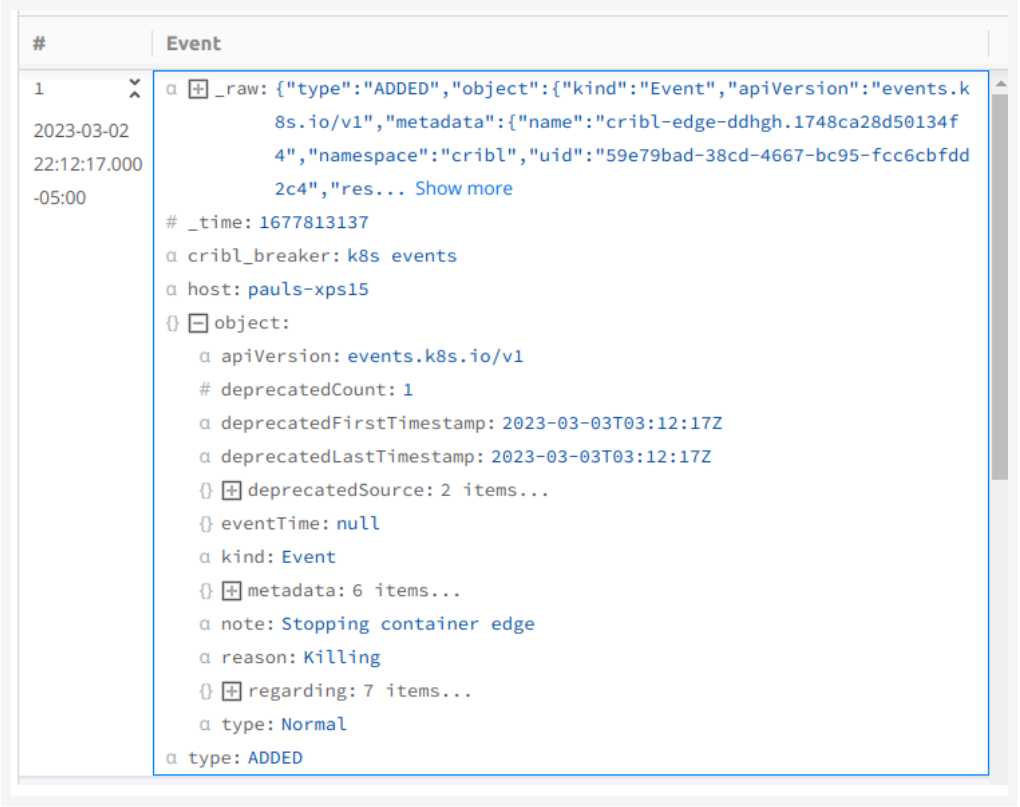
Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

Event Structure Details

This Source emits a `_raw` property as a JSON-encoded object, as other Sources do. However, there are additional `object` and `type` properties from that extraction. The logic behind this is that `timestamp` and other fields had to be extracted to keep track of starting points.

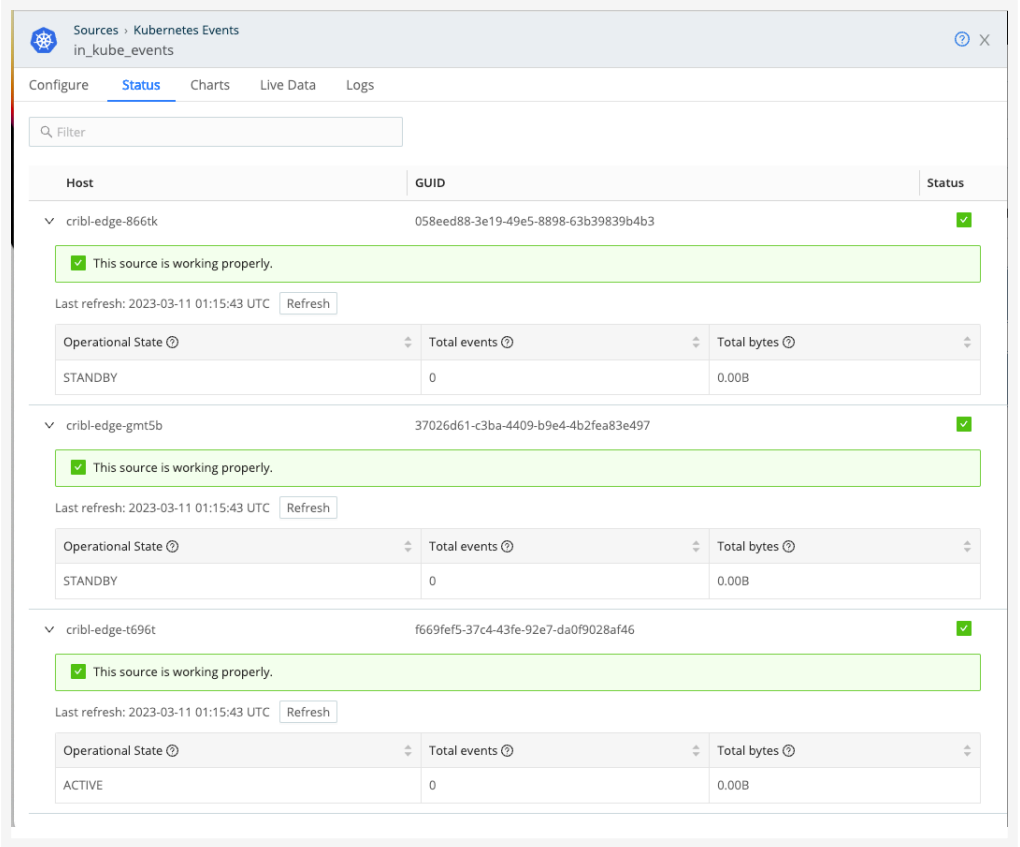
If you are sending the events over a metered connection, then you will get three copies of the same data: `_raw`, `__raw`, and the extracted `object` fields. Consider dropping the redundant fields before sending them to your Destination(s).



Sample Event

Source Operational State

To check the Source's operational state, go to **Status** and expand the host details. The **Operational State** Column shows either an **Active** state or **Standby**. **Active** indicates the Source is running or won the election. **Standby** means it's waiting to be re-elected and not currently running.



Operational State

Source Live Data

To see a sample of the Source's data, select the **Live Data** tab.

12.2.1.5. Kubernetes Logs Source

The Kubernetes Logs Source collects logs from containers on a Kubernetes node. Optionally, you can filter and enrich incoming logs. You must authorize this Source to access the Pods in all namespaces. For details, see RBAC for Kubernetes Logs Source.

Type: **System and Internal** | TLS Support: **N/A** | Event Breaker Support: **Yes** | Uses Key-Value Store

You can't configure custom Kubernetes Logs Sources.

Discover and Filter Kubernetes Pods to Monitor

The Kubernetes Logs Source connects to the Kubernetes Cluster API for metadata and to the kubelet API on the local node to discover Pods. The connection happens on a configurable **Polling interval**. The Source then runs the Pods through the **Filter Rules** to determine which ones to report on.

If no rules are configured, or if all of the rules evaluate to `true`, the Source generates logs for the Pod's containers. Conversely, if any of the rules evaluate to `false`, the Source skips the Pod's containers, and does not generate any logs for it.

The Kubernetes Logs Source can also collect logs from long-running sidecar containers (available from Kubernetes v1.33), but not init or other short-running containers.

To visualize and analyze these Kubernetes logs in more detail, consider using the Kubernetes Explorer UI in Cribl Edge. You can preview logs by navigating to the **Logs** view, filtering logs, and defining event breakers.

Cluster/Daemonset Best Practices

For the Kubernetes Logs Source to work as designed, Cribl recommends deploying Cribl Edge as a Daemonset. For details, see Deploying via Kubernetes.

This Source will refuse to run when it connects to the cluster API instead of to the `kubelet` on the local node. This happens when you run Cribl Edge outside of the cluster; or when you deploy it inside the cluster, but in a way that it can't detect the local Pod/node. In these cases, Edge will emit an error when it initializes this Source.

Configure Cribl Edge to Collect Kubernetes Logs

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Collect**. Select **Add Source** and select the Source you want from the list, then choose **Select Existing**.
    - To configure via the Routes, select **More** > **Sources**. Select the Source you want.
2. Select the default `in_kube_logs` to open the configuration modal.
3. Configure the following under **General Settings**:
    - **Input ID**: This is prefilled with the default value `in_kube_logs`, which cannot be changed via the UI, due to the single-Source restrictions above.
    - **Description**: Optionally, enter a description.
4. Next, you can configure the following **Optional Settings**:
    - **Polling interval**: How often, in seconds, to collect metrics. If not specified, defaults to `15` seconds. For considerations, see Polling Interval and Log Rotation below.
    - **Filter Rules**: Optionally, specify the Kubernetes Pods that this Source should parse to generate logs. If you specify no restrictive rules here, the Source will emit all events.
    - **Filter expression**: JavaScript expression to filter Kubernetes Pod objects. For details, see Filter Expression below.
    - **Enable timestamps**: When toggled on, Cribl Edge prefixes a timestamp to each line of the container's raw console output. When you enable timestamps, you must select the `kubernetes_logs` pre-processing Pipeline. This Pipeline removes conflicting timestamps prepended by the Source. This combination is designed for working with containers whose console output lacks proper timestamps (either because the timestamps are missing altogether or because they specify time only but not date).
    - **Tags**: Optionally, add tags that you can use to filter and group Sources in Cribl Edge's UI. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
5. Optionally, configure any Processing, Disk Spooling, and Advanced settings, or Connected Destinations outlined in the sections below.
6. Select **Save**, then **Commit & Deploy**.



The Kubernetes Logs Source only collects logs from running containers, which means it can miss logs from short-lived containers or those that exit before they start running.

Polling Interval and Log Rotation

The Kubernetes Logs Source polls the kubelet on the local node. This operation is lightweight and has minimal performance impact. However, the polling interval is crucial for capturing all log data, especially when dealing

with high log volumes and frequent log rotations.

During each poll, the Source not only discovers new Pods, but also checks existing log streams for stalls, which can occur due to container runtime log rotations. If a log rotation happens more than once between polls, one or more rotated log files will be missed, resulting in log loss.

To prevent log loss, set the **Polling interval** value to shorter than the log rotation time. For example, given a `10` second rotation interval, a polling interval closer to `5` seconds, would be more appropriate. This is because if two rotations occur within a single polling window, log data can be lost.



While Kubernetes 1.29 mitigates log rotation issues, polling intervals still matter. Long intervals can cause missed logs, especially from short-lived containers that start and stop between polls. Consider adjusting the **containerLogMaxSize** setting to prevent excessive log rotations by Kubernetes itself. This can help reduce the frequency of rotations and lessen the chance of missing logs due to polling intervals.

Filter Expression

Filters are based on the Kubernetes Pod Object definition, and evaluated from top to bottom. The first expression evaluating to `false` excludes the Pod from collection. The default filter is `!metadata.namespace.startsWith('kube-')`, which ignores Pods in the kube-* namespace.

Other **Filter expressions** examples include:

- Collect logs from Pods on a specific Node - `spec.nodeName == 'node1'`
- Ignore all DaemonSets - `metadata.ownerReferences[0].kind != 'DaemonSet'`
- Ignore Pods with specific Container names - `spec.containers[0].name != 'edge'`
- **Description**: Optional description of the rule.



Pods are either included or excluded entirely based on the Filter Rules.

Processing Settings Event Breakers

This section defines event breaking rulesets that will be applied, in order, on the `/raw` endpoint.

Event Breaker rulesets: A list of event breaking rulesets that will be applied to the input data stream before the data is sent through the Routes. Defaults to `System Default Rule`.

Event Breaker buffer timeout: How long (in milliseconds) the Event Breaker will wait for new data to be sent to a specific channel, before flushing out the data stream, as-is, to the Routes. Minimum `10` ms, default `10000`

(10 sec.), maximum `43200000` (12 hours).

Fields

In this section, you can define new fields or modify existing ones using JavaScript expressions, similar to the Eval function.

- The **Field Name** can either be a new field (unique within the event) or an existing field name to modify its value.
- The **Value** is a JavaScript expression (enclosed in quotes or backticks) to compute the field's value (can be a constant). Select this field's advanced mode icon (far right) if you'd like to open a modal where you can work with sample data and iterate on results.

This flexibility means you can:

- Add new fields to enrich the event.
- Modify existing fields by overwriting their values.
- Compute logic or transformations using JavaScript expressions.

Pre-Processing

Select a **Pipeline** (or Pack) from the drop-down to process this Source's data. Required to configure this Source via **Data Routes**; optional to configure via **Collect**/**QuickConnect**.

Disk Spooling

Enable disk spooling to search for spooled Kubernetes logs directly from Edge Nodes, without needing to forward specific logs to a Destination for storage first.

Data buckets are partitioned into subdirectories for Pods and containers, but it will be faster to search for your data by Pods.

Enable disk spooling: Determine whether or not to save Kubernetes logs to disk. When set to **Yes**, the configuration fields below become available.

Bucket time span: The amount of time that data is held in each bucket before it's written to disk. The default is 10 minutes (`10m`). Kubernetes logs are high-volume, so consider the time span you enter here (as a long time span could cause the disk to fill up quickly).

Data size limit: This is the maximum amount of disk space that spooled data is allowed to consume. Once data reaches this amount, Edge will delete data starting with the oldest buckets.

Data age limit: The duration of time for which Cribl Edge will retain data. Edge will delete data that exceeds the max age entered in this field. Example values are `2h`, `4d`. The default value is 24 hours (`24h`).

Compression: The codec that Edge uses to compress the data. The default is `gzip`.

Cribl Edge will write events to the default location:

`$CRIBL_SPOOL_DIR/in/kube_logs/{inputId}/{timeBucket}/{namespace}/{pod}/{container}`

By default, By default, `$CRIBL_SPOOL_DIR` points to `$CRIBL_HOME/state/spool`.

To change the base spool path, use the `CRIBL_SPOOL_DIR` environment variable.

Advanced Settings

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Enable load balancing: To handle high volumes of Kubernetes logs, enable load balancing. This improves the events per second (EPS) ingest rate, and is helpful when you need to ingest more Kubernetes logs EPS than a single Worker Process can handle.

When you enable this toggle, you should also increase the number of Worker Processes in Fleet Settings. The number of Worker Processes for load balancing depends on your Kubernetes log ingest volume. To ingest 40k EPS, try starting with two or four Worker Processes, then increase if needed.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

Internal Fields

Cribl Edge uses a set of internal fields to assist in forwarding data to a Destination.

These are the full objects from the Kubernetes API, showing the namespace, node, and Pod where the log originated:

- `__channel`
- `__extractedTime`
- `__final`

- `__isBroken`
- `__kube_namespace`
- `__kube_node`
- `__kube_pod`
- `__raw`
- `__source`
- `_raw`
- `_time`

For details, see Objects in Kubernetes.

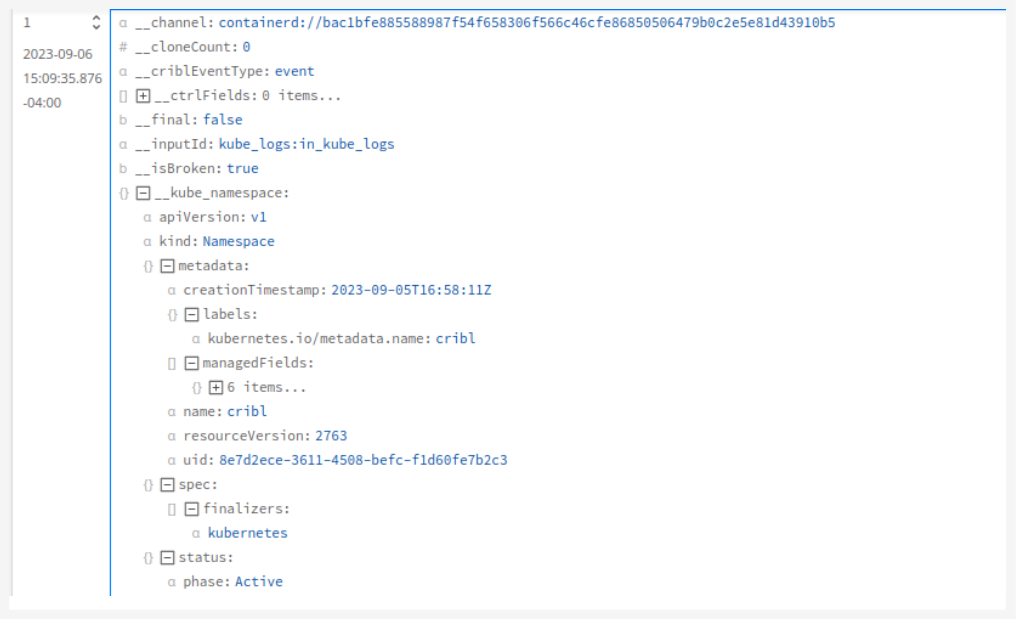
Key-Value Store

The Kubernetes Logs Source uses key-value store for container information. Container IDs are used as.

Sample Log Event

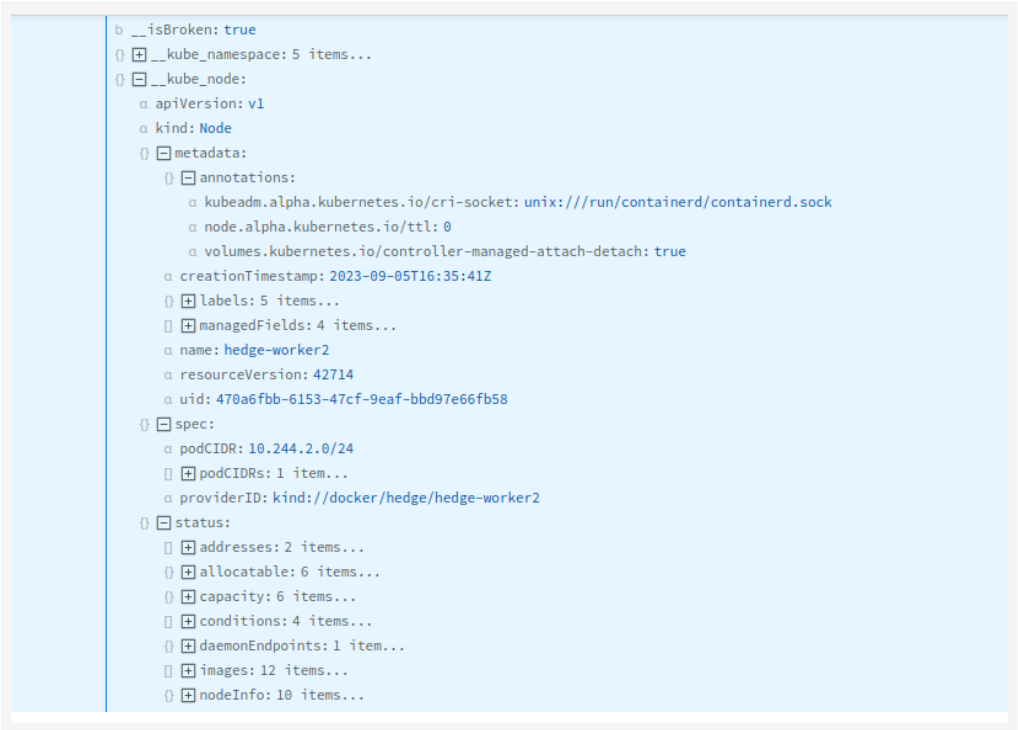
The following screenshots show the `__kube` objects corresponding to a sample log event.

Namespace Metadata



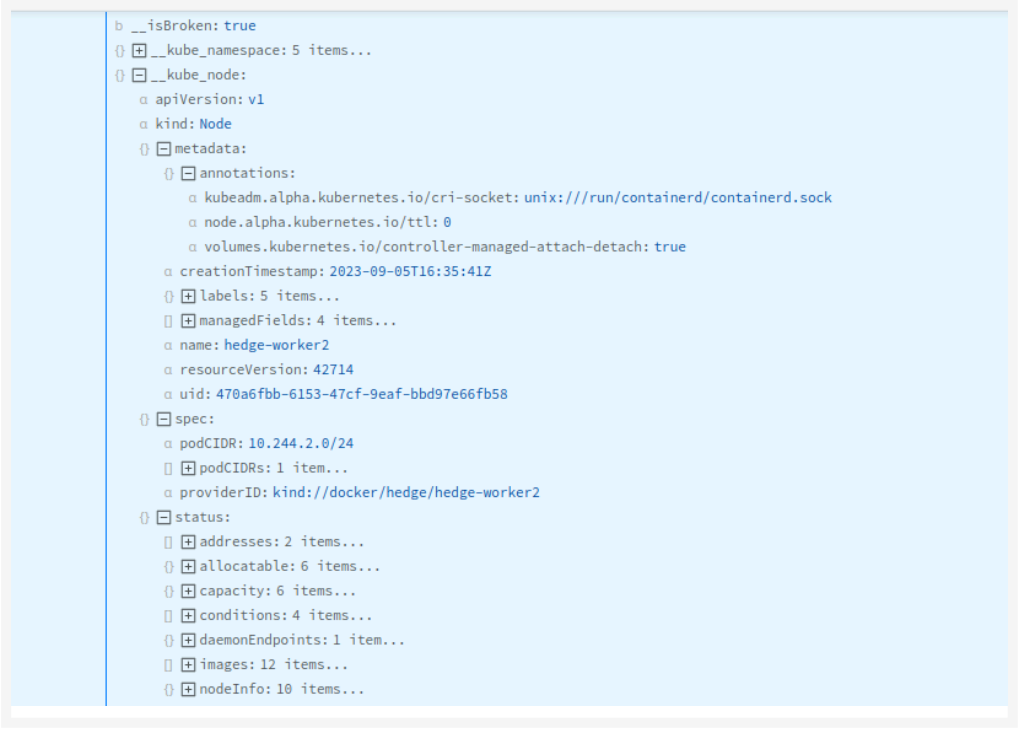
Namespace Metadata

Node Metadata



Node Metadata

Pod Metadata



Pod Metadata

12.2.1.6. Kubernetes Metrics Source

The Kubernetes Metrics Source generates events periodically based on the status and configuration of a Kubernetes cluster and its nodes, pods, and containers. For metrics details, see Kubernetes Metrics Details.

For a sample of the events this Source emits, see Live Data.

You must authorize this Source to read metric information from many resources in your cluster. For details, see RBAC for Kubernetes Metrics Source.

Type: **System and Internal** | TLS Support: **N/A** | Event Breaker Support: **No**

You can't configure custom Kubernetes Metrics Sources.

Discover and Filter Kubernetes Objects to Monitor

The Kubernetes Metrics Source connects to the Kubernetes API and loads the lists of K8s nodes, Pods, and containers in the cluster, on a configurable **Polling interval**. The Source then runs the lists through the **Filter Rules** to determine what to report on.

If no rules are configured, or if all of the rules evaluate to `true`, the Source generates metrics for the object. Conversely, if any of the rules evaluates to `false`, the Source skips the object, and does not generate any metrics for it.

Cluster/Daemonset Best Practices

For the Kubernetes Metrics Source to work as designed, Cribl recommends deploying Cribl Edge as a Daemonset. For details, see Deploying via Kubernetes.

This Source can generate redundant events when running inside the cluster and it isn't able to determine which DaemonSet it's in or which Node it's on. This happens when you run Cribl Edge inside the cluster, but in a way that it can't detect the local Pod/node. In these cases, Edge will emit an error when it initializes this Source.

Configure Cribl Edge to Collect Kubernetes Metrics

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:

- To configure via QuickConnect, navigate to **Collect**. Select **Add Source** and select the Source you want from the list, then choose **Select Existing**.
- To configure via the Routes, select **More** > **Sources**. Select the Source you want.

2. Select the default `in_kube_metrics` to open the configuration modal.
3. Configure the following under **General Settings**:
   - **Input ID**: This is prefilled with the default value `in_kube_metrics`, which cannot be changed via the UI, due to the single-Source restrictions above.
   - **Description**: Optionally, enter a description.
4. Next, you can configure the following **Optional Settings**:
   - **Polling interval**: How often, in seconds, to collect metrics. If not specified, defaults to `15s`.
   - **Filter Rules**: Optionally, specify the Kubernetes objects which this Source should parse to generate events. If you specify no restrictive rules here, the Source will emit all events.
   - **Filter expression**: JavaScript expression to filter Kubernetes objects. For details, see Filter Expression below.
   - **Description**: Optional description of the rule.
   - **Tags**: Optionally, add tags that you can use to filter and group Sources in Cribl Edge's UI. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
5. Optionally, configure any Processing, Disk Spooling, and Advanced settings, or Connected Destinations outlined in the sections below.
6. Select **Save**, then **Commit & Deploy**.

Filter Expression

Filters are based on the Kubernetes Pod Object definition, and evaluated from top to bottom. The first expression evaluating to `false` excludes the Pod from collection. The default filter is `!metadata.namespace.startsWith('kube-')`, which ignores Pods in the kube-* namespace.

Other **Filter expressions** examples include:

- Collect metrics from Pods on a specific Node - `spec.nodeName == 'node1'`
- Ignore all DaemonSets - `metadata.ownerReferences[0].kind != 'DaemonSet'`
- Ignore Pods with specific Container names - `spec.containers[0].name != 'edge'`

Processing Settings Fields

In this section, you can define new fields or modify existing ones using JavaScript expressions, similar to the Eval function.

- The **Field Name** can either be a new field (unique within the event) or an existing field name to modify its value.
- The **Value** is a JavaScript expression (enclosed in quotes or backticks) to compute the field's value (can be a constant). Select this field's advanced mode icon (far right) if you'd like to open a modal where you can work with sample data and iterate on results.

This flexibility means you can:

- Add new fields to enrich the event.
- Modify existing fields by overwriting their values.
- Compute logic or transformations using JavaScript expressions.

Pre-Processing

Select a **Pipeline** (or Pack) from the drop-down to process this Source's data. Required to configure this Source via **Data Routes**; optional to configure via **Collect**/**QuickConnect**.

Disk Spooling

Enable disk persistence: Whether to save metrics to disk. Toggle on to expose this section's remaining fields.

Bucket time span: The amount of time that data is held in each bucket before it's written to disk. The default is 10 minutes (`10m`).

Max data size: Maximum disk space the persistent metrics can consume. Once reached, Cribl Edge will delete older data. Example values: `420 MB`, `4 GB`. Default value: `100 MB`.

Max data age: How long to retain data. Once reached, Cribl Edge will delete older data. Example values: `2h`, `4d`. Default value: `24h` (24 hours).

Compression: Optionally compress the data before sending. Defaults to `gzip` compression. Select `none` to send uncompressed data.

Path location: Path to write metrics to. Default value is `$CRIBL_HOME/state/kube_metrics`.

Advanced Settings

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Connected Destinations

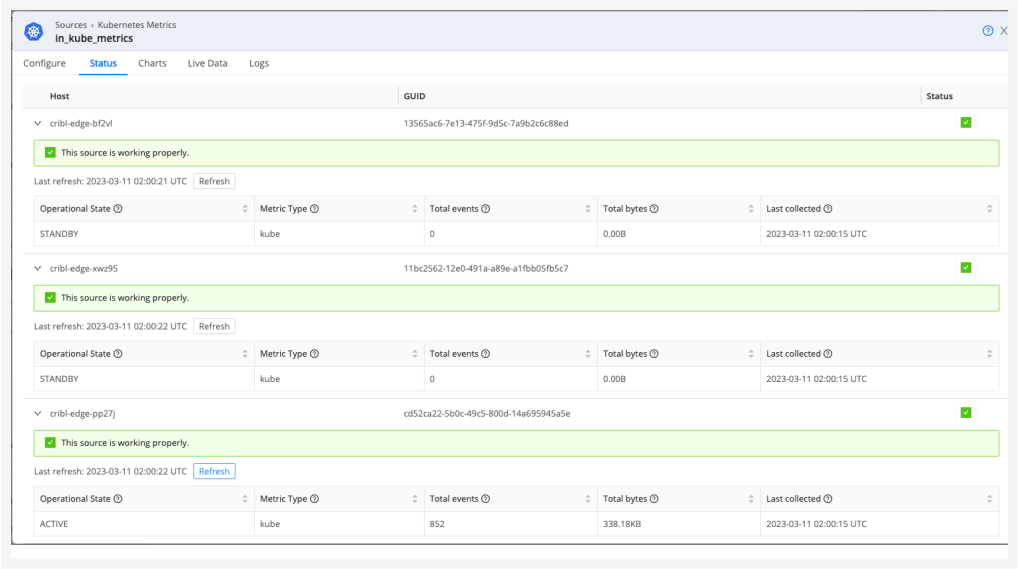
Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

Source Operational State

To avoid collecting the same metrics from multiple nodes, the Kubernetes Metrics Source uses an election method to collect metrics from only one node. The election happens every 5 minutes on all of the nodes, and the oldest node in the cluster wins.

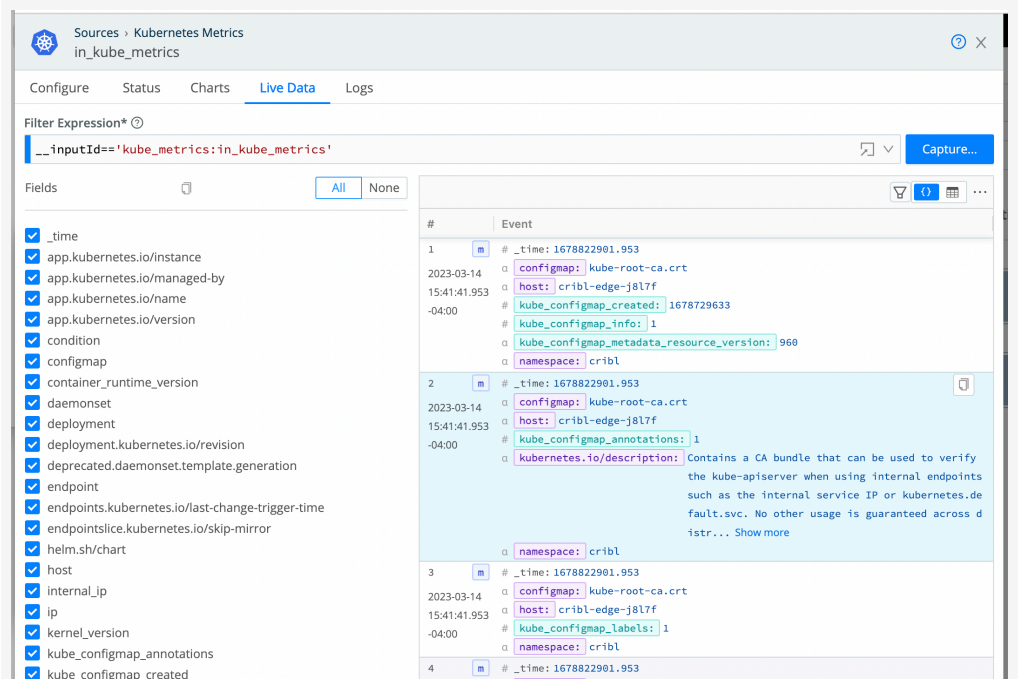
To check which node is being polled, go to **Status** and expand the host details. The **Operational State** Column shows either an **Active** state or **Standby**. **Active** indicates the host is running or won the election. **Standby** means it's waiting to be re-elected and not currently running.



Operational State

Source Live Data

To see a sample of the Source's data, select the **Live Data** tab.


Sources › Kubernetes Metrics
in_kube_metrics
Configure
Status
Charts
Live Data
Logs
Filter Expression*
__inputId=='kube_metrics:in_kube_metrics'
Capture...
Fields
All
None
_time
app.kubernetes.io/instance
app.kubernetes.io/managed-by
app.kubernetes.io/name
app.kubernetes.io/version
condition
configmap
container_runtime_version
daemonset
deployment
deployment.kubernetes.io/revision
deprecated.daemonset.template.generation
endpoint
endpoints.kubernetes.io/last-change-trigger-time
endpointslice.kubernetes.io/skip-mirror
helm.sh/chart
host
internal_ip
ip
kernel_version
kube_configmap_annotations
kube_configmap_created
#
Event
1
2023-03-14 15:41:41.953 -04:00
_time: 1678822901.953
configmap: kube-root-ca.crt
host: cribl-edge-j8l7f
kube_configmap_created: 1678729633
kube_configmap_info: 1
kube_configmap_metadata_resource_version: 960
namespace: cribl
2
2023-03-14 15:41:41.953 -04:00
_time: 1678822901.953
configmap: kube-root-ca.crt
host: cribl-edge-j8l7f
kube_configmap_annotations: 1
kubernetes.io/description: Contains a CA bundle that can be used to verify the kube-apiserver when using internal endpoints such as the internal service IP or kubernetes.default.svc. No other usage is guaranteed across distr... Show more
namespace: cribl
3
2023-03-14 15:41:41.953 -04:00
_time: 1678822901.953
configmap: kube-root-ca.crt
host: cribl-edge-j8l7f
kube_configmap_labels: 1
namespace: cribl
4
_time: 1678822901.953

12.2.1.7. System Metrics Source

Cribl Edge instances can collect metrics from their Linux hosts, and can populate some standard metrics dashboards right out of the box. To see all of the metrics this Source supports, check out Linux System Metrics Details.

Type: **System** | TLS Support: **N/A** | Event Breaker Support: **No**

Cribl Edge Workers support this System Metrics Source only when running on Linux. To collect metrics from Windows hosts, use the Windows Metrics Source.

This Source is available in Cribl.Cloud on customer-managed hybrid Workers, but not on Cribl-managed Workers in Cribl.Cloud.

Configure Cribl Edge to Collect System Metrics

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Source** and select the Source you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via Routes, select **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). Select the Source you want. Next, select **Add Source**.
2. Configure the following under **General Settings**:
    - **Enabled**: Toggle on to enable the Source.
    - **Input ID**: Enter a unique name. The default Source is prefilled with the value `in_system_metrics`, which can't be changed via the UI. If you clone this Source, Cribl Edge will add `-CLONE` to the original **Input ID**.
    - **Description**: Optionally, enter a description.
3. Next, you can configure the following **Optional Settings**:
    - **Polling interval**: How often to collect metrics, in seconds. Defaults to `10`.
    - **Tags**: Optionally, add tags that you can use to filter and group Sources in Cribl Edge's UI. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

4. Optionally, you can configure metrics for Host, Process, Container, or adjust Processing, Advanced settings and Connected Destinations outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

Host Metrics

Use the buttons to select a level of detail.

- **Basic** enables minimal metrics, averaged or aggregated.
- **All** enables full, detailed metrics, specified for individual CPUs, interfaces, and so on.
- **Custom** displays sub-menus and buttons from which you can choose a level of detail (**Basic**, **All**, **Custom**, or **Disabled**) for each type of event.
- **Disabled** means that no metrics will be generated.

The meaning of **All** and **Disabled** are self-evident. **Basic** and **Custom** have different meanings depending on event type, as follows:

System

Basic level captures load averages, uptime, and CPU count.

Custom toggles **Process** metrics on or off; these are metrics for the numbers of processes in various states.

CPU

Basic level captures active, user, system, idle, iowait percentages over all CPUs.

Custom toggles the following on or off: **Per CPU metrics**, **Detailed metrics** (meaning, for all CPU states), and **CPU time metrics** (meaning raw, monotonic CPU time counters).

Memory

Basic level captures captures total, used, available, `swap_free`, and `swap_total`.

Custom toggles **Detailed metrics** on or off. Detailed means for all memory states.

Network

Basic level captures bytes, packets, errors, connections over all interfaces.

Custom exposes the following:

- The **Interface filter**, which specifies which network interfaces to include or exclude (all are included if the filter is empty).
- **Per-interface metrics**, which toggle on or off.
- **Detailed metrics**, which toggle on or off. If on, the **Protocol metrics** toggle appears, allowing you to choose whether to generate metrics for ICMP, ICMPMsg, IP, TCP, UDP, and UDPLite.

Disk

Basic level captures disk used in percent, bytes read and written, and read and write operations, over all mounted disks.

Custom exposes the following:

- The **Device filter**, which specifies which block devices to include or exclude (all are included if the filter is empty). Wildcards and ！(not) operators are supported.
- The **Mountpoint filter**, which specifies which filesystem mountpoints to include or exclude (all are included if the filter is empty). Wildcards and ！(not) operators are supported.
- The **Filesystem type filter**, which specifies which filesystem types to include or exclude (all are included if the filter is empty). Wildcards and ！(not) operators are supported.
- **Per-device metrics**, which toggle on or off.
- **Detailed metrics**, which toggle on or off. If on, the **Enable inode metrics** toggle appears, allowing you to choose whether to generate metrics for filesystem inodes.

Process Metrics

With Process Metrics enabled, Cribl Edge captures process-specific metrics from Linux servers and reports them as events. This allows you to monitor specific processes on Cribl.Cloud instances. You can generate events for any process object.

To collect a process metrics event, create a Process Set and add a filter expression. Processes that match the filter are returned as individual events. See Collecting Metrics. Process Sets are separate from aggregate and host-wide metrics.

Process-specific metrics are **not** affected by the **Host Metrics** detail setting.

Adding a Process Set

To add a Process Set:

1. Open the System Metrics Source and access the **Configure** tab.
2. Select the **Process Metrics** menu, then **Add Process Set**.

3. Configure the details:
    - **Set Name**: The name for this process set.
    - **Filter Expression**: The JavaScript expression that will filter the processes.
    - **Include Child Processes**: When toggled on, the processes that match the filter include metrics for child processes.

Filtering Processes

You can filter processes using the field names and values from each process object.

> **Tip**
>
> To see all processes currently running on an Edge Node, select **Explore** in the submenu and open the Processes Tab.

Example filters

This filter looks for Java processes in the sleep state (an interruptible wait) that are using more than 100% of the CPU or more than 25% of memory:

```
cmdline.args[0] == 'java' && status == 'S' && (cpu > 100 || memory_percent > 25)
```

This filter looks for running NGINX web servers that have more than 12 open file descriptors:

```
service == 'nginx' && fds > 12
```

Here are some more Linux processes you could find metrics for:

- `cmdline.args[0].endsWith('/java') && cmdline.args[1] == 'nifi'`
- `exe.path == '/usr/bin/sshd'`
- `uid == 2`
- `cgroup.0 == /user.slice/user-1000.slice/`
- `stat.status == R`
- `stat.starttime > 943517`
- `cpu > 30`
- `memory_percent > 30`
- `stat.num_threads > 10`

The **Filter expression** field expects a specific syntax in order to apply the filter to the processes. These expressions evaluate to either `true` or `false`. See Build Custom Logic to Route and Process Your Data: Filter Expressions for more information.

Collecting Metrics

Once you have at least one Process Set created and saved, Cribl Edge will begin collecting process metrics at the interval defined in the **Polling interval** field (**General Settings** menu).

To view the status of the Collector and total events collected, click the **Status** tab in the Linux Metrics Source.

To view a live capture of individual metrics gathered from processes that match the Process Set, click the **Live Data** tab. The Process Set that owns each metric is represented by the `__process_set` internal field.

Supported Processes

To view the complete table of supported Linux process metrics, their descriptions, types, and dimensions, see Process-Specific Metrics.

Using Advanced Mode

To test your filter expression against a sample input, click the **Advanced mode** button within the **Filter expression** field. The **Filter expression** modal will open. See Test and Validate Expressions in Advanced Mode for more information.

Container Metrics

Use the buttons to select a level of detail.

- **Basic** generates the server event and per running container events.
- **All** adds per-device and detailed metrics.
- **Custom** provides controls for customizing which containers to generate metrics from.
- **Disabled** means that no metrics will be generated.

The **Custom** buttons displays additional controls, as follows:

- **Container filters**: Enter a filter expression to govern which containers to generate metrics from. Leave empty (the default) to generate metrics from all containers.
- **All containers**: Toggle on to include stopped and paused containers.
- **Per-device metrics**: Toggle on to generate separate metrics for each device.

- **Detailed metrics**: Toggle on to generate full container metrics.

The **Basic**, **All**, and **Custom** buttons provide the following **Advanced Settings**, which are different from those in the main **Advanced Settings** tab:

- **Docker socket**: Enter the full path(s) for Docker's UNIX domain socket. Defaults to `/var/run/docker.sock` and `/run/docker.sock`.
- **Docker timeout**: Timeout, in seconds, for the Docker API. Defaults to `5`.

Processing Settings Fields

In this section, you can define new fields or modify existing ones using JavaScript expressions, similar to the Eval function.

- The **Field Name** can either be a new field (unique within the event) or an existing field name to modify its value.
- The **Value** is a JavaScript expression (enclosed in quotes or backticks) to compute the field's value (can be a constant). Select this field's advanced mode icon (far right) if you'd like to open a modal where you can work with sample data and iterate on results.

This flexibility means you can:

- Add new fields to enrich the event.
- Modify existing fields by overwriting their values.
- Compute logic or transformations using JavaScript expressions.

Pre-Processing

If desired, choose a **Pipeline** from the drop-down if you want to process data from this Source before sending it through the Routes.

Disk Spooling

Enable disk spooling: Whether to save metrics to disk. When toggled on, exposes this section's remaining fields.

Bucket time span: The amount of time that data is held in each bucket before it's written to disk. The default is 10 minutes (`10m`).

Data size limit: Maximum disk space the persistent metrics can consume. Once reached, Cribl Edge will delete older data. Example values: `420 MB`, `4 GB`. Default value: `100 MB`.

Data age limit: How long to retain data. Once reached, Cribl Edge will delete older data. Example values: `2h`, `4d`. Default value: `24h` (24 hours).

Data compression format: Optionally compress the data before sending. Defaults to `gzip` compression. Select `none` to send uncompressed data.

Path location: Path to write metrics to. Default value is `$CRIBL_HOME/state/system_metrics`.

Advanced Settings

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

Populating Dashboards with System Metrics

Cribl has configured a Prometheus dashboard to display Cribl Edge System Metrics, and shared the dashboard in the Grafana library, which is public. This is a relatively simple dashboard suitable for showing aggregate metrics. Try this dashboard if you prefer **Basic** mode for most of your metrics.

Another useful dashboard is the one that's commonly used with Prometheus and their Node Exporter agent, found here. This dashboard can handle the highly detailed metrics. To use this dashboard, attach the `prometheus_metrics` pre-processing pipeline, and choose **All** mode.

For details on how populate your Grafana Cloud dashboards with system metrics, see System Metrics to Grafana.

12.2.1.8. System State Source

The System State Source collects snapshots of the host system's current state, on a configurable schedule, and sends them out as events to downstream systems for operational and security analytics.

Type: **System** | TLS Support: **N/A** | Event Breaker Support: **No**

This Source is available in Cribl.Cloud on customer-managed hybrid Workers, but not on Cribl-managed Workers in Cribl.Cloud. Cribl Edge on macOS offers support for only a subset of System State collectors (see Cribl Edge on MacOS).

Configure Cribl Edge to Collect System State Events

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Source** and select the Source you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via Routes, select **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). Select the Source you want. Next, select **Add Source**.
2. Configure the following under **General Settings**:
    - **Enabled**: Toggle on to enable the Source.
    - **Input ID**: Enter a unique name. The default Source is prefilled with the value `in_system_state`, which can't be changed via the UI. If you clone this Source, Cribl Edge will add `-CLONE` to the original **Input ID**.
    - **Description**: Optionally, enter a description.
3. Next, you can configure the following **Optional Settings**:
    - **Polling interval**: How often to collect metrics, in seconds. Defaults to `300`. Each run of the state collector generates events with identical `_time` values.
    - **Tags**: Optionally, add tags that you can use to filter and group Sources in Cribl Edge's UI. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
4. Optionally, you can adjust the Collector, Processing, and Advanced settings, or Connected Destinations outlined in the sections below.

5. Select **Save**, then **Commit & Deploy**.

Collector Settings

Cribl Edge contains the following System State collectors:

- Host Info
- Disks and File Systems
- DNS
- Firewall
- Hosts File
- Interfaces
- Listening Ports
- Logged-In Users
- Routes
- Services
- Users and Groups

Host Info

With the **Enabled** toggle on (the default), Cribl Edge will create events based on entries collected from the hosts file.

A partial list of Host Info events includes:

- `host`
- `hostOs` (when Cribl Edge is running in a container)
- `timestamp`
- `cribl.version`
- `cribl.mode`
- `cribl.group`
- `cribl.config_version`
- `os.arch`
- `os.cpu_count`

See the **Explore** > **System State** > **Host Info** UI for a full list of events.

The System State Source provides live system information, with most tabs updating periodically based on the **Polling Interval** configured in the Source. However, the data in the **Host Info** tab is collected once when the Source is started and does not update dynamically.

Disks and File Systems

With the **Enabled** toggle on (the default), Cribl Edge will collect entries on physical disks, partitions, and mounted filesystems. These entries can help you monitor drive status, including available space, percent utilization, inode availability, wait times, and number of blocks free.

On Cribl Edge, this option is available for Windows and Linux. You must create a different Fleet for each OS. For details see, Mapping Edge Nodes to Fleets.

For each physical disk, Cribl Edge will collect:

- name
- size (in blocks)
- blockSize (in bytes)
- model
- serial
- firmware
- partitions (count)
- interface type

For each partition on a physical disk, Cribl Edge will collect:

- disk (name)
- disk ID (number)
- type (primary or logical)
- label
- offset
- size (in blocks)
- blockSize (in bytes)
- filesystem (mount point)

For each mounted filesystem, Cribl Edge will collect:

- mount point (path)
- disk (name)
- partition (ID)
- filesystem type
- size (in blocks)
- blockSize (in bytes)
- blocksFree
- inodes
- inodesFree
- flags



Mount point refers to the directory where a filesystem is currently mounted. This information reflects the active filesystems, not entries in the `/etc/fstab` file.

DNS

With the **Enabled** toggle on (the default), Cribl Edge will create events for DNS resolvers and search entries. The event field mapping depends on the operating system.

When Cribl Edge is deployed in a container, the **DNS** Collector does not collect any events.

Linux

Cribl Edge emits events with the following fields:

Event Field	Description
`nameServer`	Name server entries from the file.
`search`	Search entries from the file.
`source`	The source of the data used to populate the event.

Windows (Cribl Edge Only)

Cribl Edge on Windows emits events with the following fields:

Event Field	Description
`ifAlias`	Interface alias name (if present).
`ifIndex`	Interface index (if present).
`addressFamily`	IPv4 or IPv6.
`nameServer`	Array of name servers.
`search`	Array of search suffixes .
`source`	The source of the data used to populate the event.
`dnsClient`	Reports `MSFT_DnsClient` when the native WMI method is used (default). Reports `Get-DnsClient` when the **Use Windows Tools** toggle is enabled in **Advanced Settings**.

Hosts File

With the **Enabled** toggle on (the default), Cribl Edge will collect entries from the `hosts` file and emit them as events.

Examples of entries in the `hosts` file are:

- `ip`: IP address of the host.
- `hostnames`: list of hostnames per IP address.

Firewall

With the **Enabled** toggle on (the default), Cribl Edge will collect events from host's defined firewall rules.

Chain Policy/References Rule(s) Event Fields

Cribl Edge emits events with the following fields:

- `chain`: Name.
- `policy`: Policy name.
- `pkts`: Total bytes packets processed by the chain.
- `bytes`: Total bytes processed by the chain.
- `references`: Number of chains referencing the current chain.

Chain Rule(s) Event Fields

Cribl Edge emits events with the following fields:

- `chain`: Name.
- `num`: Rule sequence number.
- `pkts`: Number of matched messages processed.
- `bytes`: Cumulative packet size processed (bytes).
- `target`: If the message matches the rule, the specified target is executed.
- `prot`: The specified protocol can be one of `tcp`, `udp`, `udplite`, `icmp`, `icmpv6`, `esp`, `ah`, `sctp`, `mh`, or the special keyword `all`.
- `opt`: Rarely used, this column is used to display IP options.
- `in`: Inbound network interface.
- `out`: Outbound network interface.
- `src`: The source IP address or subnet of the traffic, the latter being anywhere.
- `dest`: The destination IP address or subnet of the traffic, or anywhere.
- `match`: `ctstate`, `state`, `ADDRTYPE` info added to a state field.
- `comment`: Comment defined on the rule.
- `prot_family`: `IPv4` or `IPv6`.

Windows (Cribl Edge Only)

On Cribl Edge Windows, this collector emits events with the following fields:

- `ruleName`: Rule Name.
- `displayName`: Display Name.
- `direction`: Specifies which direction of traffic to match with this rule. The acceptable values for this parameter are `Inbound` or `Outbound`.
- `enabled`: `True` or `False`.
- `action`: Specifies the action to take on traffic that matches this rule. The acceptable values for this parameter are `Allow` or `Block`.
- `group`: Specifies the rule group (if applicable).
- `icmpType`: Specifies the `ICMP` type used.
- `policyStoreSource`: Contains a path to the policy store where the rule originated.

- `profile`: Profile conditions associated with a rule.
- `protocol`: protocols associated with firewall and `IPsec` rules.
- `localPort`: Rules for local-only port mapping.
- `remotePort`: Rules for remote port mapping.

Interfaces

With the **Enabled** toggle on (the default), Cribl Edge will create events for each of the host's network interfaces.

Cribl Edge will create separate events for each entry. All events will have identical timestamp values.

Linux IPV4/IPV6

On Linux, Cribl Edge creates `interface` events with the following fields:

- `ifIndex`: Interface index identification numbers.
- `ifName`: The name of the network interface.
- `flags`: Flags.
- `mtu`: Maximum transmission unit (MTU) in bytes for packets sent on this interface.
- `operstate`: Operational State.
- `linkType`: Physical link type.
- `macAddress`: Media Access Control address.
- `broadcast` Broadcast address.
- `addrInfo`: Array of `{family, ipAddress, mask, prefix}`.

Windows (Cribl Edge Only): IPV4/IPV6

On Windows, Cribl Edge creates `interface` events with the following fields:

- `ifIndex`: Interface index identification numbers.
- `interface`: The name of the network interface.
- `mtu`: Maximum transmission unit (MTU) in bytes for packets sent on this interface.
- `flags`: Flags.
- `macAddress`: Media Access Control address.
- `addrInfo`: Array of `{family, ipAddress, mask, prefix}`.

Listening Ports

With the **Enabled** toggle on (the default), Cribl Edge will create events from the list of listening ports and their associated process identifier (pid).

Cribl Edge will create separate events for each entry. All events will have identical timestamp values.

Sample entries include:

- `Protocol Family`: `ipv4` or `ipv6`.
- `Protocol`: `TCP`, `UDP`, or `UNIX`.
- `Socket`: `${addr}:${port}`, or `${path}` for UNIX sockets.
- `Process id`: The process identifier.
- `Program`: The program listening on the port.
- `listeningPorts`: In Cribl Edge (Windows), reports `MSFT_NetTCPConnection` and `MSFT_NetUDPEndpoint` when the native WMI method is used (default). Reports `Get-NetTCPConnection` and `Get-NetUDPEndpoint` when the **Use Windows Tools** toggle is enabled in **Advanced Settings**.

Logged-In Users

With the **Enabled** toggle on (the default), Cribl Edge will collect entries from currently logged-in users.

Linux

Cribl Edge emits events with the following fields:

- `uid`: User ID.
- `Last Login date`: Last login date.
- `terminal`: Type of the terminal device.
- `hostname`: Display the hostname of the system, if the user is connected from a remote computer.
- `userName`: Logon name of the user.

Windows (Cribl Edge Only)

Cribl Edge Windows emits events with the following fields:

- `userName`: Login name of the user.
- `sessionName`: `rdp` (Remote desktop/terminal services login session) or `console` (Direct login session).

- `id`: Session ID.
- `state`: `active` (Active session) or `disc` (Disconnected/inactive session).
- `logonTime`: Date and time the user logged on.
- `source`: Set to `Query User`.

Routes

With the **Enabled** toggle on (the default), Cribl Edge will collect entries from host's network routes and emit them as events.

Sample entries include:

- `address_family`: `ipv4` or `ipv6`.
- `destination`: The destination network or host.
- `prefix`: Emits only from Linux IPv6.
- `flags`: Flags.
- `gateway`: The gateway address.
- `interface`: Interface to which packets for this route will be sent.
- `ifIndex`: Interface ID. Emits only from Windows.
- `mask`: The netmask for the destination net. Emits only from Linux IPv4.
- `metric`: The distance to the target.
- `sequence`: Sequence number in the table.
- `source`: The source of the data used to populate the event.
- `routes`: In Cribl Edge on Windows, reports as `MSFT_NetRoute` when the native WMI method is used (default). Reports `Get-NetRoute` when the **Use Windows Tools** toggle is enabled in **Advanced Settings**.

Services

With the **Enabled** toggle on (the default), Cribl Edge will create events for each configured service (for example `systemd` and `initd`) along with their enabled and running status.

Sample entries include:

- `Name`: Name of the service.
- `Unit Activation Status`: The high-level unit activation state, i.e. generalization of `SUB` for `systemctl`. For `sysV` system value can be `enabled` or `disabled`.
- `Unit Load Status`: Reflects whether the unit definition was properly loaded. (Only `systemctl`).

- `Sub Unit Activation Status`: The low-level unit activation state, values depend on unit type.
- `Description`: Description of the service.
- `windowsService`: In Cribl Edge on Windows, reports as `Win32_Service` when the native WMI method is used (default). Reports `Get-Service` when the **Use Windows Tools** toggle is enabled in **Advanced Settings**.

When Cribl Edge is deployed in a container, the **Services** Collector does not collect any events.

Users and Groups

With the **Enabled** toggle on (the default), Cribl Edge will create events for users and groups.

Linux

The Linux collector reports exclusively on local users and groups defined within `/etc/passwd` and `/etc/group`. It does not include users or groups from external directory services like LDAP.

On Linux, Cribl Edge creates `user` events with the following fields:

- `username`
- `fullname`
- `loginShell`
- `userHome`
- `userid`
- `primaryGroup` (name of primary group, not group ID)
- `groups` (list of group names)

Cribl Edge also creates `group` events with the following fields:

- `name`
- `groupId`
- `users`

Windows (Cribl Edge Only)

Cribl Edge on Windows reports exclusively on local users and groups. On Windows hosts that are members of a domain (but are not domain controllers), Cribl Edge will only list local accounts. On Windows domain controllers, the collector will not list any users or groups as the local account list is replaced by the domain accounts.

On Windows, Cribl Edge creates `user` events with the following fields:

- `userName`
- `fullname`
- `description`
- `groups`

Cribl Edge also creates `group` events with the following fields:

- `name`
- `description`
- `users`

Processing Settings Fields

In this section, you can define new fields or modify existing ones using JavaScript expressions, similar to the Eval function.

- The **Field Name** can either be a new field (unique within the event) or an existing field name to modify its value.
- The **Value** is a JavaScript expression (enclosed in quotes or backticks) to compute the field's value (can be a constant). Select this field's advanced mode icon (far right) if you'd like to open a modal where you can work with sample data and iterate on results.

This flexibility means you can:

- Add new fields to enrich the event.
- Modify existing fields by overwriting their values.
- Compute logic or transformations using JavaScript expressions.

Pre-Processing

Select a **Pipeline** (or Pack) from the drop-down to process this Source's data. Required to configure this Source via **Data Routes**; optional to configure via **Collect**/**QuickConnect**.

Disk Spooling

Enable disk spooling: Whether to save metrics to disk. When toggled on, exposes this section's remaining fields.

Bucket time span: The amount of time that data is held in each bucket before it's written to disk. The default is 10 minutes (`10m`).

Max data size: Maximum disk space the persistent metrics can consume. Once reached, Cribl Edge will delete older data. Example values: `420 MB`, `4 GB`. Default value: `100 MB`.

Max data age: How long to retain data. Once reached, Cribl Edge will delete older data. Example values: `2h`, `4d`. Default value: `24h` (24 hours).

Compression: Currently locked to `none`. Cribl plans to add (optional) disk-spooling compression in a future release.

Path location: Path to write metrics to. Default value is `$CRIBL_HOME/state/system_state`.

Advanced Settings

Use Windows Tools: Toggled off by default. This legacy setting will be removed in a future release. We highly recommend keeping this option disabled and using the newer native capabilities which are faster and more reliable.

This setting affects Windows Edge Nodes only, it has no effect on Linux Edge Nodes or in Cribl Stream.

- If toggled off, the System State Source interacts directly with the Windows Management Instrumentation (WMI) API, resulting in faster data collection.
- If toggled on, the System State Source will collect events using Powershell cmdlets.

Use legacy collection for LastLog: Toggled off by default. This legacy setting will be removed in a future release. We highly recommend keeping this option disabled and using the newer native capabilities which are faster and more reliable.

Toggle on if you encounter issues with collecting LastLog information to revert to the legacy collection method.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing

12.2.2. Internal

You can use the following Sources to receive data from internal senders.

Source	Description
Cribl HTTP	Receive data from peer Nodes managed by the same Leader when raw TCP traffic is not an option
Cribl TCP	Send data between Worker Nodes/Edge Nodes connected to the same Leader
Cribl Internal	Capture Edge's internal logs and metrics and send them through Routes and Pipelines
Datagen	Generate data from datagen files

12.2.2.1. Cribl HTTP Source

The Cribl HTTP Source receives data from a Cribl HTTP Destination. Relaying data between a Cribl HTTP Destination and Cribl HTTP Source pair prevents double-billing: You won't incur any additional charges or credit usage after initial ingest to the first Fleet (depending on your Leader setup - see Transfer Data Between Workspaces or Environments for more details).

It's common to send data from Cribl Edge to a Cribl Stream Worker Group using this kind of pairing.

Type: **Internal** | TLS Support: **YES** | Event Breaker Support: **No**

For guidance on when to choose this Source versus the Cribl TCP Source, see Interoperation Protocols.

Within Cribl Stream, the Cribl HTTP Source is available only in Distributed deployments. In single-instance mode, or for testing, you can use the Raw HTTP/S Source instead. However, this substitution will not facilitate sending all internal fields, as described below.

Looking for the Cribl HTTP Source in **Cribl Search**? See Ingest Cribl Stream/Edge Data into Cribl Search.

How It Works

You can use the Cribl HTTP Source to transfer data between Edge Nodes and Workers. If the Cribl HTTP Source receives data from its Cribl HTTP Destination counterpart on another Worker, you're billed for ingress only once - when Cribl first receives the data.

Using a Cribl HTTP Destination/Source pair, you're not charged for subsequent data sent to:

- Workers that share a Leader.
- Workers that don't share a Leader, but are configured in Cribl.Cloud Workspaces within the same Organization.
- Workers that do not share a Leader, but that do have the exact same on-prem license installed.

This use case is common in hybrid Cribl.Cloud deployments, where a customer-managed (hybrid) Worker Node sends data to a Worker in Cribl.Cloud for additional processing and routing to Destinations. However, the Cribl HTTP Destination/Source pair can similarly reduce your metered data ingress in other scenarios, such as on-prem Edge to on-prem Stream.

As one usage example, assume that you want to send data from one Node deployed on-prem, to another that is deployed in Cribl.Cloud. You could do the following:

- Create an on-prem File System Collector (or whatever Collector or Source is suitable) for the data you want to send to Cribl.Cloud.
- Create an on-prem Cribl HTTP Destination.
- Create a Cribl HTTP Source, on the target Stream Worker Group or Edge Fleet in Cribl.Cloud.
- For an on-prem Node configure a File System Collector to send data to the Cribl HTTP Destination, and from there to the Cribl HTTP Source in Cribl.Cloud.
- On Cribl-managed Nodes in Cribl.Cloud, make sure that TLS is either disabled on both the Cribl HTTP Destination and the Cribl HTTP Source it's sending data to, or enabled on both. Otherwise, no data will flow. On Cribl.Cloud instances, the Cribl HTTP Source ships with TLS enabled by default.

Configuration Requirements

The key points about configuring this architecture are:

- The Destination's **Cribl endpoint** field must point to the **Address** and **Port** you've configured on its peer Cribl HTTP Source(s).
- Cribl 3.5.4 was a breakpoint in Cribl HTTP Leader/Worker communications. Nodes running the Cribl HTTP Source on Cribl 3.5.4 and newer can send data only to Nodes running v.3.5.4 and newer. Nodes running the Cribl HTTP Source on Cribl 3.5.3 and older can send data only to Nodes running v.3.5.3 and older.

Configure the Cribl HTTP Source

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Source** and select the Source you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). Select the Source you want. Next, select **Add Source**.
2. In the **New Source** modal, configure the following under **General Settings**:
    - **Enabled**: Toggle on to enable the Source.
    - **Input ID**: Enter a unique name to identify this Cribl HTTP Source definition. If you clone this Source, Cribl Edge will add `-CLONE` to the original **Input ID**.
    - **Description**: Optionally, enter a description.
    - **Address**: Enter the address to bind on. Defaults to `0.0.0.0` (all addresses).
    - **Port**: Enter the port number to listen on, for example, `10200`.

3. Next, you can configure the following **Optional Settings**:
    - **Tags**: Optionally, add tags that you can use for filtering and grouping in the Cribl Edge UI. Use a tab or hard return between (arbitrary) tag names. These tags aren't added to processed events.
4. Optionally, configure any TLS settings, Persistent Queue, Processing, Advanced settings and Connected Destinations outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

TLS Settings (Server Side)



On Cribl.Cloud, you can enable TLS using Cribl-provided certficates if desired.

Enabled Default is toggled off. When toggled on, exposes this section's remaining fields.

Certificate name: Select a predefined certificate from the drop-down. A **Create** button is available to create a new certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

Passphrase: Passphrase to use to decrypt private key.

Certificate path: Server path containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

CA certificate path: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

Authenticate client (mutual auth): Require clients to present their certificates. Used to perform mutual authentication using SSL certs. Default is toggled off. When toggled on, exposes these two additional fields:

- **Validate client certificates**: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (for example, the system's CA). Default is toggled off.
- **Common name**: Regex matching subject common names in peer certificates allowed to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. For example, to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

Enable Persistent Queue: Defaults to toggled off. When toggled on:

Mode: Choose a mode from the drop-down:

- With `Smart` mode, PQ will write events to the filesystem only when it detects backpressure from the processing engine.
- With `Always On` mode, PQ will always write events directly to the queue before forwarding them to the processing engine.

Buffer size limit: Maximum number of events to hold in-memory before dumping them to disk.

Commit frequency: Number of events to send before committing that Stream has read them.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space the queue is allowed to consume. Once this limit is reached, Cribl Edge stops queueing and applies the fallback **Queue-full behavior**. Enter a numeral with units of KB, MB, etc.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this value, Cribl Edge will append `/<worker-id>/<output-id>`.

Compression: Codec to use to compress the persisted data, once a file is closed. Defaults to `None`; `Gzip` is also available.

Processing Settings Fields

In this section, you can define new fields or modify existing ones using JavaScript expressions, similar to the Eval function.

- The **Field Name** can either be a new field (unique within the event) or an existing field name to modify its value.
- The **Value** is a JavaScript expression (enclosed in quotes or backticks) to compute the field's value (can be a constant). Select this field's advanced mode icon (far right) if you'd like to open a modal where you can work with sample data and iterate on results.

This flexibility means you can:

- Add new fields to enrich the event.
- Modify existing fields by overwriting their values.
- Compute logic or transformations using JavaScript expressions.

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline or Pack to process data from this input before the data is sent through the Routes.

Auth Token

This section is only available for Cribl.Cloud environments. It is intended for use when migrating data from an on-prem environment to a Cribl.Cloud environment.

If you previously had an on-prem deployment of Cribl and you are moving forward with a Cloud-first or Cloud only solution, you can use connected environments to onboard your data from on-prem Edge Nodes to their counterpart Cribl.Cloud Edge Nodes. See Migrate Data from On-Prem to Cribl.Cloud for implementation details.

Advanced Settings

Show originating IP: Toggle on when clients are connecting through a proxy that supports the `X-Forwarded-For` header to keep the client's original IP address on the event instead of the proxy's IP address. This setting affects how the Source handles the `__srcIpPort` field.

Capture request headers: Toggle on to add request headers to events, in the `__headers` field.

Health check endpoint: Toggle on to enable a health check endpoint specific to this Source, `http(s)://<host>:<port>/cribl_health`. A `200` HTTP response code is returned when the Source is healthy. Otherwise, two errors you could receive are:

- `ECONNRESET` where the Source failed to initialize due to not having listeners on the port.
- `503` or `Server is busy, max active connections reached` indicate there are too many connections per Worker Process.

Active request limit (on-prem only): Maximum number of active requests allowed for this Source, per Worker Process. Defaults to `256`. Enter `0` for unlimited.

Activity log sample rate: Determines how often request activity is logged at the `info` level. The default `100` value logs every 100th value; a `1` value would log every request; a `10` value would log every 10th request; etc.

Requests-per-socket limit: The maximum number of requests Cribl Edge should allow on one socket before instructing the client to close the connection. Defaults to `0` (unlimited). See Balancing Connection Reuse Against Request Distribution below.

Socket timeout (seconds): How long Cribl Edge should wait before assuming that an inactive socket has timed out. The default `0` value means wait forever.

Request timeout (seconds): How long to wait for an incoming request to complete before aborting it. The default `0` value means wait indefinitely.

Keep-alive timeout (seconds): After the last response is sent, Cribl Edge will wait this long for additional data before closing the socket connection. Defaults to `5` seconds; minimum is `1` second; maximum is `600` seconds (10 minutes).



The longer the **Keep-alive timeout**, the more Cribl Edge will reuse connections. The shorter the timeout, the closer Cribl Edge gets to creating a new connection for every request. When request frequency is high, you can use longer timeouts to reduce the number of connections created, which mitigates the associated cost.

IP allowlist regex: Grants access to requests originating from specific IP addresses that match a defined pattern. Unmatched requests are rejected with a 403 (Forbidden) status code. Defaults to `.*` (allow all).

IP denylist regex: Blocks requests originating from specific IP addresses that match a defined pattern, even if they would be allowed by default. Rejected requests receive a 403 (Forbidden) status code. Defaults to `^$` (allow all).

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Balancing Connection Reuse Against Request Distribution

Requests-per-socket limit allows you to limit the number of HTTP requests an upstream client can send on one network connection. Once the limit is reached, Cribl Edge uses HTTP headers to inform the client that it must establish a new connection to send any more requests. (Specifically, Cribl Edge sets the HTTP `Connection` header to `close`.) After that, if the client disregards what Cribl Edge has asked it to do and tries to send another HTTP request over the existing connection, Cribl Edge will respond with an HTTP status code of `503 Service Unavailable`.

Use this setting to strike a balance between connection reuse by the client, and distribution of requests among one or more Edge Node processes by Cribl Edge:

- When a client sends a sequence of requests on the same connection, that is called connection reuse. Because connection reuse benefits client performance by avoiding the overhead of creating new connections, clients have an incentive to **maximize** connection reuse.
- Meanwhile, a single process on that Edge Node will handle all the requests of a single network connection, for the lifetime of the connection. When receiving a large overall set of data, Cribl Edge performs better when the workload is distributed across multiple Edge Node processes. In that situation, it makes sense to **limit** connection reuse.

There is no one-size-fits-all solution, because of variation in the size of the payload a client sends with a request and in the number of requests a client wants to send in one sequence. Start by estimating how long connections will stay open. To do this, multiply the typical time that requests take to process (based on payload size) times the number of requests the client typically wants to send.

If the result is 60 seconds or longer, set **Requests-per-socket limit** to force the client to create a new connection sooner. This way, more data can be spread over more Edge Node processes within a given unit of time.

For example: Suppose a client tries to send thousands of requests over a very few connections that stay open for hours on end. By setting a relatively low **Requests-per-socket limit**, you can ensure that the same work is done over more, shorter-lived connections distributed between more Edge Node processes, yielding better performance from Cribl Edge.

A final point to consider is that one Cribl Edge Source can receive requests from more than one client, making it more complicated to determine an optimal value for **Requests-per-socket limit**.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

Internal Fields

Cribl Edge uses a set of internal fields to assist in handling of data. These "meta" fields are **not** part of an event, but they are accessible, and Functions can use them to make processing decisions.

The Cribl HTTP Source (and the Cribl TCP Source) treat internal fields differently than other Sources do. That's because of the difference in the way that incoming data originates.

Other Sources ingest data that's not coming from Cribl Edge or Stream, meaning that no Cribl internal fields can be present in that data when it arrives at the Source, and the Source is free to add internal fields without clobbering (overwriting) anything that existed already.

By contrast, the Cribl HTTP Source and the Cribl TCP Source ingest data that's coming from a Cribl HTTP or Cribl TCP Destination. That data **can** contain internal fields when it arrives at the Source. This means that if the Source adds internal fields, those could potentially clobber what existed before.

To avoid this problem, the Cribl HTTP Source and the Cribl TCP Source add a unique `__forwardedAttrs` (i.e., "forwarded attributes") field. The nested structure of the `__forwardedAttrs` field contains any of the following fields that are present in the arriving data:

Internal Fields
`__headers` - Added only when **Advanced Settings** > **Capture request headers** is toggled on.
`__inputId`
`__outputId`
`fleet` / `group`
`__srcIpPort` - See details below.

Other Fields
`cribl_breaker`
`cribl_pipe`

These fields are **copied** into `__forwardedAttrs`, not moved there. As the data (apart from `__forwardedAttrs`) moves through the Source and any Pipelines, the values of these fields can be overwritten. But the copies of these fields in `__forwardedAttrs` remain unchanged, so you can retrieve them as necessary.

Overriding `__srcIpPort` with Client IP/Port

The `__srcIpPort` field's value contains the IP address and (optionally) port of the HTTP client sending data to this Source.

When any proxies (including load balancers) lie between the HTTP client and the Source, the last proxy adds an `X-Forwarded-For` header whose value is the IP/port of the original client. With multiple proxies, this header's value will be an array, whose first item is the original client IP/port.

If `X-Forwarded-For` is present, and **Advanced Settings** > **Show originating IP** is toggled off, the original client IP/port in this header will override the value of `__srcIpPort`.

If **Show originating IP** is toggled on, the `X-Forwarded-For` header's contents will **not** override the `__srcIpPort` value. (Here, the upstream proxy can convey the client IP/port without using this header.)

Troubleshooting

Dropping request because token invalid",“authToken”: “Bas...Njc=”

The specified token is invalid. Note that the above message is logged only at the debug level.

12.2.2.2. Cribl TCP Source

The Cribl TCP Source receives data from a Cribl TCP Destination. Relaying data between a Cribl TCP Destination and Cribl TCP Source pair prevents double-billing: You won't incur any additional charges or credit usage after initial ingest to the first Fleet (depending on your Leader setup - see Transfer Data Between Workspaces or Environments for more details).

It's common to send data from Cribl Edge to a Cribl Stream Worker Group using this kind of pairing.



Type: **Internal** | TLS Support: **YES** | Event Breaker Support: **No**

For guidance on when to choose this Source versus the Cribl HTTP Source, see Interoperation Protocols.

Within Cribl Stream, the Cribl TCP Source is available only in Distributed deployments. In single-instance mode, or for testing, you can use the TCP JSON Source instead. However, this substitution will not facilitate sending all internal fields, as described below.

How It Works

You can use the Cribl TCP Source to transfer data between Edge Nodes and Workers. If the Cribl TCP Source receives data from its Cribl TCP Destination counterpart on another Worker, you're billed for ingress only once - when Cribl first receives the data.

Using a Cribl HTTP Destination/Source pair, you're not charged for subsequent data sent to:

- Workers that share a Leader.
- Workers that don't share a Leader, but are configured in Cribl.Cloud Workspaces within the same Organization.
- Workers that do not share a Leader, but that do have the exact same on-prem license installed.

This use case is common in hybrid Cribl.Cloud deployments, where a customer-managed (hybrid) Worker Node sends data to a Worker in Cribl.Cloud for additional processing and routing to Destinations. However, the Cribl TCP Destination/Source pair can similarly reduce your metered data ingress in other scenarios, such as on-prem Cribl Edge to on-prem Cribl Stream.

As one usage example, assume that you want to send data from one Node deployed on-prem, to another that is deployed in Cribl.Cloud. You could do the following:

- Create an on-prem File System Collector (or whatever Collector or Source is suitable) for the data you want to send to Cribl.Cloud.

- Create an on-prem Cribl TCP Destination.
- Create a Cribl TCP Source, on the target Cribl Stream Worker Group or Cribl Edge Fleet in Cribl.Cloud.
- For an on-prem Node, configure a File System Collector to send data to the Cribl TCP Destination, and from there to the Cribl TCP Source in Cribl.Cloud.
- On Cribl-managed Nodes in Cribl.Cloud, make sure that TLS is either disabled on both the Cribl TCP Destination and the Cribl TCP Source it's sending data to, or enabled on both. Otherwise, no data will flow. On Cribl.Cloud instances, the Cribl TCP Source ships with TLS enabled by default.

Configuration Requirements

The key points about configuring this architecture are:

- When you configure the Cribl TCP Destination, its **Address** and **Port** values must point to the **Address** and **Port** you've configured on the Cribl TCP Source.
- Cribl 3.5.4 was a breakpoint in Cribl TCP Leader/Worker communications. Nodes running the Cribl TCP Source on Cribl 3.5.4 and newer can send data only to Nodes running v.3.5.4 and newer. Nodes running the Cribl TCP Source on Cribl 3.5.3 and older can send data only to Nodes running v.3.5.3 and older.

Configure the Cribl TCP Source

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Source** and select the Source you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). Select the Source you want. Next, select **Add Source**.
2. In the **New Source** modal, configure the following under **General Settings**:
    - **Enabled**: Toggle on to enable the Source.
    - **Input ID**: Enter a unique name to identify this Cribl TCP Source definition. If you clone this Source, Cribl Edge will add `-CLONE` to the original **Input ID**.
    - **Description**: Optionally, enter a description.
    - **Address**: Enter hostname/IP to listen for TCP JSON data. For example, `localhost` or `0.0.0.0`.
    - **Port**: Enter the port number to listen on, for example, `10300`.
3. Next, you can configure the following **Optional Settings**:

- **Tags**: Optionally, add tags that you can use for filtering and grouping in the Cribl Edge UI. Use a tab or hard return between (arbitrary) tag names. These tags aren't added to processed events.

4. Optionally, configure any TLS settings, Persistent Queue, Processing, and Advanced settings, or Connected Destinations outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

TLS Settings (Server Side)



On Cribl.Cloud, you can enable TLS using Cribl-provided certficates if desired.

Enabled: Defaults to toggled off. When toggled on:

Certificate: Name of the predefined certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

Passphrase: Passphrase to use to decrypt private key.

Certificate path: Server path containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

CA certificate path: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

Authenticate client (mutual auth): Require clients to present their certificates. Used to perform mutual authentication using SSL certs. Default is toggled off. When toggled on:

- **Validate client certs**: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (for example, the system's CA). Default is toggled off.
- **Common name**: Regex matching subject common names in peer certificates allowed to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. For example, to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

Enable persistent queue: Defaults to toggled off. When toggled on:

Mode: Choose a mode from the drop-down:

- With `Smart` mode, PQ will write events to the filesystem only when it detects backpressure from the processing engine.
- With `Always On` mode, PQ will always write events directly to the queue before forwarding them to the processing engine.

Buffer size limit: Maximum number of events to hold in memory before dumping them to disk. Defaults to `1000`.

Commit frequency: Number of events to send before committing that Cribl Edge has read them. Defaults to `42`.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, and so on. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space the queue is allowed to consume. Once this limit is reached, Cribl Edge stops queueing and applies the fallback **Queue-full behavior**. Enter a numeral with units of KB, MB, and so on.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this value, Cribl Edge will append `/<worker-id>/<output-id>`.

Compression: Codec to use to compress the persisted data, once a file is closed. Defaults to `None`; `Gzip` is also available.

Processing Settings Fields

In this section, you can define new fields or modify existing ones using JavaScript expressions, similar to the Eval function.

- The **Field Name** can either be a new field (unique within the event) or an existing field name to modify its value.
- The **Value** is a JavaScript expression (enclosed in quotes or backticks) to compute the field's value (can be a constant). Select this field's advanced mode icon (far right) if you'd like to open a modal where you can work with sample data and iterate on results.

This flexibility means you can:

- Add new fields to enrich the event.
- Modify existing fields by overwriting their values.
- Compute logic or transformations using JavaScript expressions.

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline or Pack to process data from this input before the data is sent through the Routes.

Auth Token

This section is only available for Cribl.Cloud environments. It is intended for use when migrating data from an on-prem environment to a Cribl.Cloud environment.

If you previously had an on-prem deployment of Cribl and you are moving forward with a Cloud-first or Cloud only solution, you can use connected environments to onboard your data from on-prem Edge Nodes to their counterpart Cribl.Cloud Edge Nodes. See Migrate Data from On-Prem to Cribl.Cloud for implementation details.

Advanced Settings

Enable proxy protocol: Toggle on if the connection is proxied by a device that supports proxy protocol v1 or v2.

Enable load balancing (Cribl Stream only): Toggle on to distribute the data from incoming TCP connections across multiple Worker Processes. This improves performance compared to the default of having all data from an incoming connection processed on a single Worker Process.

- This spins up an extra Worker Process (named `wLB` in **Settings** > **Processes**) on the Worker Node, which handles splitting data from incoming TCP connections across all other Worker Processes.
- When enabled, the option to include common fields in the JSON header is not supported. This means fields specified under `fields` in the JSON header are not automatically added to all events.



Load balancing is available in Cribl Stream single-instance and Distributed deployments when a TCP port is configured.

Active connection limit: Maximum number of active connections allowed per Worker Process. Defaults to `1000`. Set a lower value if connection storms are causing the Source to hang. Set to `0` for unlimited connections.

Socket idle timeout (seconds): The duration that Cribl Edge will wait for activity on an idle TCP socket before closing the connection. Disabled when set to `0`, the default.

Forced socket termination timeout (seconds): The extra time the server waits before forcibly closing a socket that has been idle (**TCP socket idle timeout**) or exceeded its maximum lifespan (**TCP socket max lifespan**) but has not yet properly closed. This prevents resource leaks caused by unresponsive clients or network issues. Configure based on network latency and client behavior. Default: `30` seconds. Set to `0` to disable.

Socket max lifespan (seconds): The duration that a socket is allowed to remain open, regardless of activity. This setting prevents resource exhaustion (such as TCP pinning) by limiting the lifespan of connections.

Configure based on expected connection durations and resource availability. Disabled when set to `0`, the default.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the configuration will be enabled everywhere.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

Internal Fields

Cribl Edge uses a set of internal fields to assist in handling of data. These "meta" fields are **not** part of an event, but they are accessible, and Functions can use them to make processing decisions.

The Cribl TCP Source (and the Cribl HTTP Source) treat internal fields differently than other Sources do. That's because of the difference in the way that incoming data originates.

Other Sources ingest data that's not coming from Cribl Edge or Cribl Stream, meaning that no Cribl internal fields can be present in that data when it arrives at the Source, and the Source is free to add internal fields without clobbering (overwriting) anything that existed already.

By contrast, the Cribl TCP Source and the Cribl HTTP Source ingest data that's coming from a Cribl TCP or Cribl HTTP Destination. That data **can** contain internal fields when it arrives at the Source. This means that if the Source adds internal fields, those could potentially clobber what existed before.

To avoid this problem, the Cribl TCP Source and the Cribl HTTP Source add a unique `__forwardedAttrs` (that is, "forwarded attributes") field. The nested structure of the `__forwardedAttrs` field contains any of the following fields that are present in the arriving data:

Internal Fields
`__srcIpPort`
`__inputId`
`__outputId`
`fleet / group`

Other Fields
`cribl_breaker`
`cribl_pipe`

These fields are **copied** into `__forwardedAttrs`, not moved there. As the data (apart from `__forwardedAttrs`) moves through the Source and any Pipelines, the values of these fields can be overwritten. But the copies of these fields in `__forwardedAttrs` remain unchanged, so you can retrieve them as necessary.

Periodic Logging

Cribl Edge logs metrics about incoming requests and ingested events once per minute.

These logs are stored in the `metrics.log` file. To view them in the UI, open the Source's **Logs** tab and choose **Worker Process X Metrics** from the drop-down, where **X** is the desired Worker Process.

This kind of periodic logging helps you determine whether a Source is in fact still healthy even when no data is coming in.

12.2.2.3. Cribl Internal Source

The Cribl Internal Source enables you to capture and send Cribl Edge's own internal logs and metrics through Routes and Pipelines.

Type: **Internal** | TLS Support: **N/A** | Event Breaker Support: **No**

You can't configure custom Cribl Internal Sources.

Scope and Purpose

In Distributed mode, this Source's CriblLogs option can process internal logs only from Worker Processes. (Logs on the Leader remain on the Leader, because the Leader Node is not part of any processing path.)

In both Distributed and Single-instance mode, this Source's CriblLogs option omits API Process logs, meaning that it omits telemetry/license-validation traffic. You can, however, use a Script Collector to check for API Server (or Worker Group) events.

This Source's CriblMetrics option offers Cribl's most up-to-date and precise view of event throughput and transformation, aggregated every 2 seconds at the Worker Process level. The CriblLogs option is less granular - tracking logs that are written once per minute - so CriblLogs might fail to reflect Worker Process crashes or restarts.

In Cribl.Cloud, on Cribl-managed Worker Groups in Cribl.Cloud, the CriblLogs internal Source contains only logs related to Sources and Destinations.

Configure Cribl Internal Logs/Metrics as a Data Source

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Source** and select the Source you want from the list, then choose **Select Existing**.
    - To configure via the Routes, select **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). Select the Source you want.
2. Next, select either the default `CriblLogs` or `CriblMetrics` to open the configuration modal.

3. Configure the **General Settings** for your selected Source.
    - For Cribl Logs, configure the following:
        - **Enabled**: This duplicates the parent page's **Enabled** toggle. Toggle on to enable Cribl logs as a Source.
        - **Input ID**: This is prefilled with the default value `CriblLogs`, which cannot be changed.
        - **Description**: Optionally, enter a description.
    - For Cribl Metrics, configure the following:
        - **Enabled**: This duplicates the parent page's **Enabled** toggle. Toggle on to enable Cribl metrics as a Source.
        - **Input ID**: This is prefilled with the default value `CriblMetrics`, which cannot be changed.
        - **Description**: Optionally, enter a description.
4. Next, you can configure the following **Optional Settings** for your selected Source:
    - For Cribl Logs, configure the following:
        - **Tags**: Optionally, add tags that you can use to filter and group Sources in Cribl Edge's UI. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
    - For Cribl Metrics, configure the following:
        - **Metric name prefix**: Enter an optional prefix that will be applied to metrics provided by Cribl Edge. The prefix defaults to `cribl.logstream.`
        - **Full fidelity**: Toggle off to reduce CPU load by excluding metrics for any fields listed in **Settings** > **Global** > **System** > **General Settings** > **Limits** > **Metrics** > **Disable field metrics**. See Controlling Metrics Volume for details. By default, these include:
            - `cribl.logstream.host.(in_bytes,in_events,out_bytes,out_events)`
            - `cribl.logstream.index.(in_bytes,in_events,out_bytes,out_events)`
            - `cribl.logstream.project.(in_bytes, in_events, out_bytes, out_events)`
            - `cribl.logstream.source.(in_bytes,in_events,out_bytes,out_events)`
            - `cribl.logstream.sourcetype.(in_bytes,in_events,out_bytes,out_events)`
    - **Tags**: Optionally, add tags that you can use to filter and group Sources in Cribl Edge's UI. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
5. Optionally, configure any Processing and Advanced settings outlined in the sections below.
6. Select **Save**, then **Commit & Deploy**.

Processing Settings Fields

In this section, you can define new fields or modify existing ones using JavaScript expressions, similar to the Eval function.

- The **Field Name** can either be a new field (unique within the event) or an existing field name to modify its value.
- The **Value** is a JavaScript expression (enclosed in quotes or backticks) to compute the field's value (can be a constant). Select this field's advanced mode icon (far right) if you'd like to open a modal where you can work with sample data and iterate on results.

This flexibility means you can:

- Add new fields to enrich the event.
- Modify existing fields by overwriting their values.
- Compute logic or transformations using JavaScript expressions.

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline or Pack to process data from this input before the data is sent through the Routes.

Advanced Settings

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

Reporting Metrics Less Frequently

By default, Cribl Edge generates internal metrics every 2 seconds. To consume metrics at longer intervals, you can use or adapt the `cribl_metrics_rollup` Pipeline that ships with Cribl Edge.

Attach this Pipeline to your **Cribl Internal** Source as a pre-processing Pipeline. The Pipeline's **Rollup Metrics** Function has a default **Time Window** of 30 seconds, which you can adjust to a different granularity as needed.

This provides a second lever to reduce granularity, in addition to the Full fidelity toggle described above.

Omitting `sourcetype`

You can easily drop the `sourcetype` attribute from metrics events, leaving only `event_sourcetype`. This will prevent duplicate `sourcetype` events from being routed to Destinations.

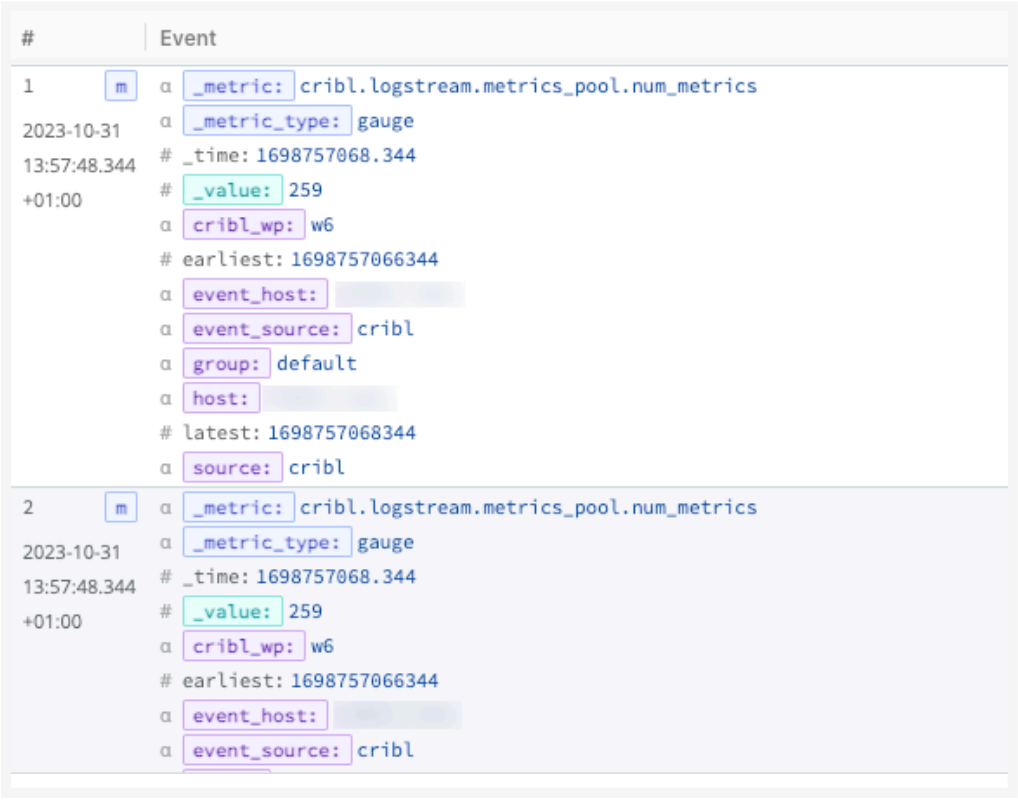
To do this: In the same `cribl_metrics_rollup` pre-processing Pipeline (or a clone) that you attach to your Source, enable the final Eval Function, which applies this **Filter** expression to remove the `sourcetype` field: `_metric && _metric.startsWith('cribl.logstream.sourcetype.')`

Duplicated Fields/Dimensions

The CriblMetrics Source operates on metrics that Edge Nodes report to their Leader Nodes. Typically included are `source` and `host` fields.

Sending metrics from this Source to Splunk is one common use case. Because Splunk might overwrite these two fields, the Source copies their values into new dimensions with added `event_` prefixes: `event_source` and `event_host`. This way, if Splunk does overwrite `source` and/or `host`, their original values remain intact in the new dimensions with `event_` prefixes.

Here's an example of how the added dimensions look in the **Live Capture** window:



Doubled fields

If you are not sending to a downstream service that overwrites `source` or `host` fields, you can use an appropriate Function to drop the added dimensions. You also have the option to suppress these fields entirely by specifying them in **Settings** > **Global** > **System** > **General Settings** > **Limits** > **Metrics** > **Disable field metrics**.

Internal Fields

The following fields will be added to all events/metrics:

- `source`: set to `cribl`.
- `host`: set to the hostname of the Cribl instance.

Use these fields to guide these events/metrics through Cribl Routes.



All Cribl internal fields are subject to change and modification. Cribl provides them to assist with analytics and diagnostics, but does not guarantee that they will remain available.

12.2.2.4. Datagen Source

Cribl Edge supports generating data from datagen files, as detailed in Using Datagens. When a datagen is enabled, each Worker Process uses the specified data generator file to generate events. These events proceed through Routes and Pipelines, or through a QuickConnect configuration, to configured Destinations. Whichever Worker Process generated an event from the file will also send the same event.



Type: **System** | TLS Support: **N/A** | Event Breaker Support: **No**

Configure Cribl Edge to Generate Sample Data

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Source** and select the Source you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). Select the Source you want. Next, select **Add Source**.
2. In the **New Source** modal, configure the following under **General Settings**:
    - **Input ID**: Enter a unique name to identify this Source definition. If you clone this Source, Cribl Edge will add `-CLONE` to the original **Input ID**.
    - **Description**: Optionally, enter a description.
    - **Datagens**: List of datagens. For details, see Datagen Fields below.
3. Next, you can configure the following **Optional Settings**:
    - **Tags**: Optionally, add tags that you can use for filtering and grouping in the Cribl Edge UI. Use a tab or hard return between (arbitrary) tag names. These tags aren't added to processed events.
4. Optionally, configure any Processing and Advanced settings outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

Datagen Fields

To configure the list of datagens, include the following:

- **Data generator file**: Name of the datagen file.

- **Events per second per Worker Node**: Maximum number of events to generate per second, per Worker Node/Edge Node. Defaults to `10`.

Processing Settings Fields

In this section, you can define new fields or modify existing ones using JavaScript expressions, similar to the Eval function.

- The **Field Name** can either be a new field (unique within the event) or an existing field name to modify its value.
- The **Value** is a JavaScript expression (enclosed in quotes or backticks) to compute the field's value (can be a constant). Select this field's advanced mode icon (far right) if you'd like to open a modal where you can work with sample data and iterate on results.

This flexibility means you can:

- Add new fields to enrich the event.
- Modify existing fields by overwriting their values.
- Compute logic or transformations using JavaScript expressions.

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline or Pack to process data from this input before the data is sent through the Routes.

Advanced Settings

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

Internal Fields

Cribl Edge uses a set of internal fields to assist in handling of data. These “meta” fields are **not** part of an event, but they are accessible, and Functions can use them to make processing decisions.

Fields for this Source:

- `__inputId`

12.2.3. Amazon

You can use the following Sources to receive data from Amazon services.

Source	Description
Amazon Data Firehose	Receive Amazon Data Firehose streams via HTTP endpoint

12.2.3.1. Amazon Data Firehose Source

Cribl Edge supports receiving data from Amazon Data Firehose delivery streams via the **HTTP endpoint** destination option.

Type: **Push** | TLS Support: **YES** | Event Breaker Support: **No**

This Source supports gzip-compressed inbound data when the `Content-Encoding: gzip` connection header is set.

AWS Configuration

Configuring the endpoint and port settings for integrating Cribl with Amazon Data Firehose varies depending on whether you're using Cribl.Cloud or a non-Cribl.Cloud deployment.

- **Cribl.Cloud**:
  - Configure the AWS destination **HTTP endpoint URL** to your Cribl.Cloud endpoint `https://default.main.<organizationId>.cribl.cloud` on port `443`. Replace `<organizationId>` with your actual Organization ID.
  - Configure the Cribl Edge Amazon Data Firehose Source to listen on port `10443` internally.
- **Non-Cribl.Cloud**:
  - Configure the AWS destination **HTTP endpoint URL** to the public URL of your Cribl Edge instance.
  - Configure the Cribl Edge Amazon Data Firehose Source to either listen on port `443` with elevated privileges or use a reverse proxy to forward traffic from port `443` to a non-privileged port, like `10443`.

Configure Cribl Edge to Receive Data over HTTP(S) from Amazon Data Firehose

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
   - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Source** and select the Source you want from the list, choosing either **Select Existing** or **Add New**.

- To configure via the Routes, select **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). Select the Source you want. Next, select **Add Source**.

2. In the **New Source** modal, configure the following under **General Settings**:
   - **Enabled**: Toggle on to enable the Source.
   - **Input ID**: Enter a unique name to identify this Source definition. If you clone this Source, Cribl Edge will add `-CLONE` to the original **Input ID**.
   - **Description**: Optionally, enter a description.
   - **Address**: Address to bind on. Defaults to `0.0.0.0` (all addresses).
   - **Port**: Enter the port number to listen on. In Cribl.Cloud, use port `10443`, as this is where the data is processed internally.
3. Under **Authentication**, enter the **Auth tokens**:
   - **Token**: Shared secrets to be provided by any client (Authorization: <token>). Click **Generate** to create a new secret. If empty, unauthorized access **will be permitted**.
4. Next, you can configure the following **Optional Settings**:
   - **Tags**: Optionally, add tags that you can use to filter and group Sources in Cribl Edge's UI. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
5. Optionally, you can adjust the TLS, Persistent Queue Settings, Processing, and Advanced settings, or Connected Destinations outlined in the sections below.
6. Select **Save**, then **Commit & Deploy**.

TLS Settings (Server Side)

Enabled: Defaults to toggled off. When toggled on:

Certificate: Name of the predefined certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

Passphrase: Passphrase to use to decrypt private key.

Certificate path: Server path containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

CA certificate path: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

Authenticate client (mutual auth): Require clients to present their certificates. Used to perform mutual authentication using SSL certs. Default is toggled off. When toggled on:

- **Validate client certificates**: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (for example, the system's CA). Default is toggled off.
- **Common name**: Regex that a peer certificate's subject attribute must match in order to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. (For example, to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.) If the subject attribute contains Subject Alternative Name (SAN) entries, the Source will check the regex against all of those but ignore the Common Name (CN) entry (if any). If the certificate has no SAN extension, the Source will check the regex against the single name in the CN.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In the **Persistent Queue Settings** tab, you can optionally specify persistent queue storage, using the following controls. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Source Persistent Queues (sPQ)

> On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the **Enable persistent queue** toggle. If enabled, PQ is automatically configured in `Always On` mode, with a maximum queue size of 1 GB disk space allocated per PQ-enabled Source, per Worker Process.
>
> The 1 GB limit is on uncompressed inbound data, and the queue does not perform any compression. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable persistent queue: Default is toggled off. When toggled on:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to the Cribl Stream data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from the Cribl Stream data processing engine.

`Smart` mode only engages when necessary, such as when a downstream Destination becomes blocked *and* the **Buffer size limit** reaches its limit. When persistent queue is set to `Smart` mode, Cribl attempts to flush the queue when every new event arrives. The only time events stay in the buffer is when a downstream Destination becomes blocked.

Buffer size limit: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. This buffer is for all connections, not just per Worker Process. For that reason, this can dramatically expand memory usage. Connections share this limit, which may result in slightly lower throughput for higher numbers of connections. For higher numbers of connections, consider increasing the limit.

Commit frequency: The number of events to send downstream before committing that Stream has read them. Defaults to `42`.

File size limit: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, and so forth. If not specified, Cribl Stream applies the default `1 MB`.

Queue size limit: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, and so forth. Can be set as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** in Group or Fleet settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file closes. Defaults to `None`; `Gzip` is also available.

In Cribl Stream 4.1 and newer, the Source persistent queue default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Optimize Source Persistent Queues (sPQ).

You can optimize Workers' startup connections and CPU load at **Group/Fleet settings** > Worker Processes.

Processing Settings Fields

In this section, you can define new fields or modify existing ones using JavaScript expressions, similar to the Eval function.

- The **Field Name** can either be a new field (unique within the event) or an existing field name to modify its value.

- The **Value** is a JavaScript expression (enclosed in quotes or backticks) to compute the field's value (can be a constant). Select this field's advanced mode icon (far right) if you'd like to open a modal where you can work with sample data and iterate on results.

This flexibility means you can:

- Add new fields to enrich the event.
- Modify existing fields by overwriting their values.
- Compute logic or transformations using JavaScript expressions.

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline or Pack to process data from this input before the data is sent through the Routes.

Advanced Settings

Show originating IP: Toggle on when clients are connecting through a proxy that supports the `X-Forwarded-For` header to keep the client's original IP address on the event instead of the proxy's IP address. This setting affects how the Source handles the `__srcIpPort` field.

Capture request headers: Toggle on to add request headers to events, in the `__headers` field.

Health check endpoint: Toggle on to enable a health check endpoint specific to this Source, `http(s)://<host>:<port>/cribl_health`. A `200` HTTP response code is returned when the Source is healthy. Otherwise, two errors you could receive are:

- `ECONNRESET` where the Source failed to initialize due to not having listeners on the port.
- `503` or `Server is busy, max active connections reached` indicate there are too many connections per Worker Process.

Active request limit (on-prem only): Maximum number of active requests allowed for this Source, per Worker Process. Defaults to `256`. Enter `0` for unlimited.



Raising this limit can increase throughput by allowing more concurrent data requests, but increases resource usage and load on both your Cribl Edge infrastructure and on downstream Destinations. Before raising this limit, ensure:

- Your Cribl Edge deployment has sufficient capacity to support higher request concurrency, including CPU, memory, or number of Worker Processes.
- Downstream Destinations are correctly sized and tuned to accept the higher data ingest rate, preventing backpressure. See Manage Backpressure for more information.

> Improper sizing on either side can result in dropped events, delayed processing, or overall system instability.

Activity log sample rate: Determines how often request activity is logged at the `info` level. The default `100` value logs every 100th value; a `1` value would log every request; a `10` value would log every 10th request; etc.

Requests-per-socket limit: The maximum number of requests Cribl Edge should allow on one socket before instructing the client to close the connection. Defaults to `0` (unlimited). See Balancing Connection Reuse Against Request Distribution below.

Socket timeout (seconds): How long Cribl Edge should wait before assuming that an inactive socket has timed out. The default `0` value means wait forever.

Request timeout (seconds): How long to wait for an incoming request to complete before aborting it. The default `0` value means wait indefinitely.

Keep-alive timeout (seconds): After the last response is sent, Cribl Edge will wait this long for additional data before closing the socket connection. Defaults to `5` seconds; minimum is `1` second; maximum is `600` seconds (10 minutes).

> The longer the **Keep-alive timeout**, the more Cribl Edge will reuse connections. The shorter the timeout, the closer Cribl Edge gets to creating a new connection for every request. When request frequency is high, you can use longer timeouts to reduce the number of connections created, which mitigates the associated cost.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Balancing Connection Reuse Against Request Distribution

Requests-per-socket limit allows you to limit the number of HTTP requests an upstream client can send on one network connection. Once the limit is reached, Cribl Edge uses HTTP headers to inform the client that it must establish a new connection to send any more requests. (Specifically, Cribl Edge sets the HTTP `Connection` header to `close`.) After that, if the client disregards what Cribl Edge has asked it to do and tries to send another HTTP request over the existing connection, Cribl Edge will respond with an HTTP status code of `503 Service Unavailable`.

Use this setting to strike a balance between connection reuse by the client, and distribution of requests among one or more Edge Node processes by Cribl Edge:

- When a client sends a sequence of requests on the same connection, that is called connection reuse. Because connection reuse benefits client performance by avoiding the overhead of creating new connections, clients have an incentive to **maximize** connection reuse.

- Meanwhile, a single process on that Edge Node will handle all the requests of a single network connection, for the lifetime of the connection. When receiving a large overall set of data, Cribl Edge performs better when the workload is distributed across multiple Edge Node processes. In that situation, it makes sense to **limit** connection reuse.

There is no one-size-fits-all solution, because of variation in the size of the payload a client sends with a request and in the number of requests a client wants to send in one sequence. Start by estimating how long connections will stay open. To do this, multiply the typical time that requests take to process (based on payload size) times the number of requests the client typically wants to send.

If the result is 60 seconds or longer, set **Requests-per-socket limit** to force the client to create a new connection sooner. This way, more data can be spread over more Edge Node processes within a given unit of time.

For example: Suppose a client tries to send thousands of requests over a very few connections that stay open for hours on end. By setting a relatively low **Requests-per-socket limit**, you can ensure that the same work is done over more, shorter-lived connections distributed between more Edge Node processes, yielding better performance from Cribl Edge.

A final point to consider is that one Cribl Edge Source can receive requests from more than one client, making it more complicated to determine an optimal value for **Requests-per-socket limit**.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

Internal Fields

Cribl Edge uses a set of internal fields to assist in handling of data. These "meta" fields are **not** part of an event, but they are accessible, and functions can use them to make processing decisions.

Fields for this Source:

- `__headers` - Added only when **Advanced Settings** > **Capture request headers** is toggled on.
- `__inputId`
- `__raw`
- `__srcIpPort` - See details below.

Records also include the following metadata specific to the Amazon Data Firehose service:

- `__firehoseArn`: The Amazon Resource Name (ARN) of the Amazon Data Firehose delivery stream that served as the data source.
- `__firehoseReqId`: A unique identifier assigned to the specific data ingestion request processed by Amazon Data Firehose.
- `__firehoseEndpoint`: The service endpoint URL of the Amazon Data Firehose service that received the data.
- `__firehoseToken`: An authentication token associated with the Amazon Data Firehose data ingestion request, used for source validation.

Overriding `__srcIpPort` with Client IP/Port

The `__srcIpPort` field's value contains the IP address and (optionally) port of the Amazon Data Firehose client sending data to this Source.

When any proxies (including load balancers) lie between the Amazon Data Firehose client and the Source, the last proxy adds an `X-Forwarded-For` header whose value is the IP/port of the original client. With multiple proxies, this header's value will be an array, whose first item is the original client IP/port.

If `X-Forwarded-For` is present, and **Advanced Settings** > **Show originating IP** is toggled off, the original client IP/port in this header will override the value of `__srcIpPort`.

If **Show originating IP** is toggled on, the `X-Forwarded-For` header's contents will **not** override the `__srcIpPort` value. (Here, the upstream proxy can convey the client IP/port without using this header.)

Limitations

If you set the optional `IntervalInSeconds` and/or `SizeInMBs` parameters in the Amazon Data Firehose `BufferingHints` API, beware of selecting extreme values (toward the ends of the API's supported ranges). These can send more bytes than Cribl Edge can buffer, causing Cribl Edge to send HTTP 500 error responses to Amazon Data Firehose.

Troubleshooting

The Source's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they are ingested. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the ingestion process, including any errors or warnings that may have occurred.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify ingestion issues. Analyze the graphs showing events and bytes in/out over time.

Common Issues

Dropping request because token invalid",“authToken”: “Bas...Njc=”

The specified token is invalid. Note that the above message is logged only at the debug level.

12.2.4. Prometheus

You can use the following Sources to receive data from Prometheus services.

Source	Description
Prometheus Scraper	Receive batched data from Prometheus targets
Prometheus Remote Write	Receive metric data from Prometheus via the remote write protocol
Grafana	Receive metric and log data from Grafana Agent via Prometheus remote write
Loki	Receive log data from Grafana Loki

12.2.4.1. Prometheus Edge Scraper Source

Cribl Edge supports receiving batched data from Prometheus targets.



Type: **Internal** | TLS Support: **No** | Event Breaker Support: **No**

This Source currently does not support Prometheus metadata.

This is an Internal (Pull) Source. To ingest Prometheus streaming data, see Prometheus Remote Write.

Configuring Cribl Edge to Scrape Prometheus Data

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Collect**. Select **Add Source** and select the Source you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **More** > **Sources**. Select the Source you want. Next, select **Add Source**.
2. Configure the following under **General Settings**:
    - **Input ID**: Enter a unique name to identify this Source definition. If you clone this Source, Cribl Edge will add `-CLONE` to the original **Input ID**.
    - **Description**: Optionally, enter a description.
    - **Discovery type**: Use this drop-down to select a discovery mechanism for targets. See Discovery Type below for the options and the resulting controls displayed. Some **Discovery type** options replace this section's **Targets** field with additional controls - while also adding an **AWS IAM** left tab to the modal.
    - **Poll interval**: Specify how often (in seconds) to scrape targets for metrics. Defaults to `15` seconds. This value must be an integer that divides evenly into `60`.
3. Next, you can configure the following **Optional Settings**:
    - **Extra dimensions**: Specify the dimensions to include in events. Defaults to `host` and `source`.
    - **Tags**: Optionally, add tags that you can use for filtering and grouping in the Cribl Edge UI. Use a tab or hard return between (arbitrary) tag names. These tags aren't added to processed events.
4. Optionally, configure any Authentication, Processing, and Advanced settings, or Connected Destinations outlined in the sections below.

5. Select **Save**, then **Commit & Deploy**.

Discovery Type

Use this drop-down to select a discovery mechanism for targets. To manually enter a targets list, use Static (the default). To enable dynamic discovery of endpoints to scrape, select DNS or AWS EC2. Each selection exposes different controls and/or tabs, listed below.

Static Discovery

The `Static` option adds a **General Settings** > **Targets** field, in which you enter a list of specific Prometheus targets from which to pull metrics. Click **Add Target** to expose a table with the following options:

- **Protocol**: Select `http` (the default) or `https` as the protocol to use when collecting metrics.
- **Host**: Specify the host name to pull metrics from.
- **Port**: Specify the port number to append to the metrics URL for discovered targets. Defaults to `9090`.
- **Path**: Specify a path to use when collecting metrics from discovered targets. Defaults to `/metrics`.

DNS Discovery

The `DNS` option adds the following extra fields to its **General Settings** tab:

- **Record type**: Select the DNS record type to resolve. Defaults to `SRV` (Service). Other options are `A` or `AAAA`.
- **DNS names**: Enter a list of DNS names to resolve.
- **Protocol**: Select `http` (the default) or `https` as the protocol to use when collecting metrics.
- **Path**: Specify a path to use when collecting metrics from discovered targets. Defaults to `/metrics`.

AWS EC2 Discovery

The `AWS EC2` option adds AWS IAM to the modal, and adds extra fields to the **General Settings** tab:

- **Protocol**: Select `http` (the default) or `https` as the protocol to use when collecting metrics.
- **Port**: Specify the port number to append to the metrics URL corresponding to discovered targets. Defaults to `9090`.
- **Path**: Specify a path to use when collecting metrics from discovered targets. Defaults to `/metrics`.
- **Region**: Select the AWS Region in which to discover EC2 instances with metrics endpoints to scrape.
- **Use public IP**: Toggle on (default) to use the public IP address for discovered targets. Toggle off to use a private IP address.

- **Search filter**: Click **Add filter** to apply filters when searching for EC2 instances. Each filter row provides two columns:
  - **Filter name**: Select standard attributes from the drop-down, or type in custom attributes.
  - **Filter values**: Enter values to match within this row's attribute, Press `Enter` between values. (If you specify no values, the search will return only `running` EC2 instances.)

Selecting `AWS EC2` also adds controls to the **Advanced Settings** tab, as described below.

Kubernetes Node

The `Kubernetes Node` option supports collecting metrics from an endpoint on the local node where Cribl Edge is running. Selecting this option adds the following extra fields to the **General Settings** tab:

- **Protocol**: Select `http` (the default) or `https` as the protocol to use when collecting metrics.
- **Port**: Specify the port number to append to the metrics URL corresponding to discovered targets. Defaults to `9090`.
- **Path**: Specify a path to use when collecting metrics from discovered targets. Defaults to `/metrics`.

Selecting `Kubernetes Node` also adds a control to the **Authentication** tab, as described below.

You must first authorize the Source to discover a Kubernetes Node. For details, see RBAC for Prometheus Edge Scraper.

Kuberenetes Pods

The `Kubernetes Pods` option supports building a list of endpoints to collect from Pods on the same node where Cribl Edge is running. This option adds the following extra fields to the **General Settings** tab - use a custom expression to set the configuration properties on each field.

- **Protocol**: Set the protocol to use when collecting metrics. Defaults to: `metadata.annotations['prometheus.io/scheme'] || 'http'`
- **Port**: Specify the port number to append to the metrics URL corresponding to discovered targets. Defaults to: `metadata.annotations['prometheus.io/port'] || '9090'`
- **Path**: Specify a path to use when collecting metrics from discovered targets. Defaults to: `metadata.annotations['prometheus.io/path'] || '/metrics'`
- **Filter rules**: Optionally, add rules to determine which Pods to discover for metrics. If no rules are provided, Cribl Edge will search all Pods. Otherwise, it will search Pods where rules' expressions evaluate to `true`. Each rule consists of an expression and an optional description.
  - **Filter expression**: JavaScript expression applied to Pods' objects. Return `true` to include it. Filters are based on the Kubernetes Pod Object definition, and are evaluated from top to bottom. The first expression evaluating to `false` excludes a Pod from collection. The default filter is

`metadata.annotations['prometheus.io/scrape']`, which scrapes the Pod if the annotation is true.

- **Description**: Optional description of the rule.

Selecting `Kubernetes Pods` also adds a control to the **Authentication** tab, as described below.

You must first authorize the Source to discover Kubernetes Pods. For details, see RBAC for Prometheus Edge Scraper.

Authentication (Prometheus)

Use the **Authentication method** drop-down to select one of these authentication options for Prometheus:

- **Manual**: In the resulting **Username** and **Password** fields, enter Basic authentication credentials corresponding to your Prometheus targets.
- **Secret**: This option exposes a **Secret** drop-down, in which you can select a stored secret that references your credentials described above. The secret can reside in Cribl Edge's internal secrets manager or (if enabled) in an external KMS. Click **Create** if you need to configure a new secret.

Authentication Settings for Kubernetes

This additional setting appears only when **General Settings** > **Discovery Type** is set to `Kubernetes Pods` or `Kubernetes Node`.

- **Kubernetes**: Select this option to use Kubernetes authentication. There are no details to configure, because Cribl Edge will authenticate using the Environment Variables or Service Account tokens mounted in the Edge Pod(s). For details, see Deploying via Kubernetes.

AWS IAM

With the AWS EC2 target discovery type, you can configure AssumeRole behavior on AWS.

Assume Role

- **Enable for EC2**: Toggle on if you want to use `AssumeRole` credentials to access EC2.
- **AssumeRole ARN**: Enter the Amazon Resource Name (ARN) of the role to assume.
- **External ID**: Enter the External ID to use when assuming the role.

AWS Authentication Options

Auto: This default option uses the AWS SDK for JavaScript to automatically obtain credentials in the following order of attempts:

1. **Environment Variables**: Loaded from environment variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`.
2. **IAM Identity Center (SSO)**: Loaded from AWS IAM Identity Center configuration.
3. **Shared Credentials File**: Loaded from the shared credentials file (`~/.aws/credentials`).
4. **IAM Roles for Amazon EC2/ECS**: Loaded from AWS Identity and Access Management (IAM) roles attached to an EC2 instance or ECS container.
5. **JSON File on Disk**: Loaded from a JSON file on disk.
6. **Other Credential-Provider Classes**: Other credential-provider classes provided by the AWS SDK for JavaScript.

The `Auto` method works both when running on AWS and in other environments where the necessary credentials are available through one of the above methods.



SSO Providers

When using the auto authentication method, you can leverage SSO providers like SAML and Okta to issue temporary credentials. These credentials should be set in the environment variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`. The AWS SDK will then use these environment variables to authenticate.

Manual: If not running on AWS, you can select this option to enter a static set of user-associated IAM credentials (your access key and secret key) directly or by reference. This is useful for Edge Nodes not in an AWS VPC, for example, those running a private cloud. This option displays:

- **Access key**: Enter your AWS access key. If not present, will fall back to the `env.AWS_ACCESS_KEY_ID` environment variable, or to the metadata endpoint for IAM role credentials.
- **Secret key**: Enter your AWS secret key. If not present, will fall back to the `env.AWS_SECRET_ACCESS_KEY` environment variable, or to the metadata endpoint for IAM credentials.

Secret: If not running on AWS, you can select this option to supply a stored secret that references an AWS access key and secret key. This option displays:

- **Secret key pair**: Use the drop-down to select a secret key pair that you've configured in Cribl Edge's internal secrets manager or (if enabled) an external KMS. Click **Create** if you need to configure a key pair.

Processing Settings Fields

In this section, you can define new fields or modify existing ones using JavaScript expressions, similar to the Eval function.

- The **Field Name** can either be a new field (unique within the event) or an existing field name to modify its value.
- The **Value** is a JavaScript expression (enclosed in quotes or backticks) to compute the field's value (can be a constant). Select this field's advanced mode icon (far right) if you'd like to open a modal where you can work with sample data and iterate on results.

This flexibility means you can:

- Add new fields to enrich the event.
- Modify existing fields by overwriting their values.
- Compute logic or transformations using JavaScript expressions.

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline or Pack to process data from this input before the data is sent through the Routes.

Disk Spooling

Enable disk spooling: Whether to save metrics to disk. When toggled on, it exposes this section's remaining fields.

Bucket time span: The amount of time that data is held in each bucket before it's written to disk. The default is 10 minutes (`10m`).

Max data size: Maximum disk space the persistent metrics can consume. Once reached, Cribl Edge will delete older data. Example values: `420 MB`, `4 GB`. Default value: `1 GB`.

Max data age: How long to retain data. Once reached, Cribl Edge will delete older data. Example values: `2h`, `4d`. Default value: `24h` (24 hours).

Compression: Defaults to `gzip`.

Cribl Edge will write metrics to the default location: `CRIBL_HOME/state/spool/in/edge_prometheus/${inputId}`. Use the environment variable `CRIBL_SPOOL_DIR`, to change the default path.

Advanced Settings

HTTP Connection Timeout: The amount of time (in milliseconds) to wait for the HTTP connection to establish before it times out. `1-60000` is the allowable range. Enter `0` to disable the timeout.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Advanced Settings for AWS

These additional settings appear only when **General Settings** > **Discovery Type** is set to `AWS EC2`.

Endpoint: Specify an EC2-compatible service endpoint. If empty, this defaults to AWS' Region-specific endpoint.

Signature version: The signature version to use for signing EC2 requests. Defaults to `v4`.

Reuse connections: Whether to reuse connections between requests. Toggling on (default) can improve performance.

Reject unauthorized certificates: Whether to reject certificates that cannot be verified against a valid Certificate Authority (for example, self-signed certificates). Defaults to toggled on, the restrictive option.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

Internal Fields

Cribl Edge uses a set of internal fields to assist in handling of data. These "meta" fields are **not** part of an event, but they are accessible, and Functions can use them to make processing decisions.

Fields for this Source:

- `__source`
- `__isBroken`
- `__inputId`
- `__final`
- `__criblMetrics`
- `__channel`
- `__cloneCount`
- `__kube_pod` when Discovery Mode is set to **Kubernetes Node**.
- `__kube_node` and `__kube_pod` when Discovery Mode is set to **Kubernetes Pods**.

Proxying Requests

If you need to proxy HTTP/S requests, see System Proxy Configuration.

Troubleshooting

Dropping request because token invalid",“authToken”: “Bas...Njc=”

The specified token is invalid. Note that the above message is logged only at the debug level.

12.2.4.2. Prometheus Remote Write Source

Cribl Edge supports receiving metric data from Prometheus instances that are configured to send data via the remote write protocol.

Type: **Push** | TLS Support: **YES** | Event Breaker Support: **No**

This Source assumes that incoming data is snappy-compressed.

Looking for the Prometheus Remote Write Source in **Cribl Search**? See Ingest Prometheus Metrics into Cribl Search.

Configure Cribl Edge to Receive Metrics from Prometheus Remote Write Sources

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Source** and select the Source you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). Select the Source you want. Next, select **Add Source**.
2. In the **New Source** modal, configure the following under **General Settings**:
    - **Enabled**: Toggle on to enable the Source.
    - **Input ID**: Enter a unique name to identify this Source definition. If you clone this Source, Cribl Edge will add `-CLONE` to the original **Input ID**.
    - **Description**: Optionally, enter a description.
    - **Address**: Enter the hostname/IP to listen to. Defaults to `0.0.0.0`.
    - **Port**: Enter the port number to listen on.
    - **Remote Write API endpoint**: Enter the absolute path on which to listen for Prometheus requests. Defaults to `/write`, which will (in this example) expand as: `http://<your-upstream-URL>:<your-port>/write`.
3. Next, you can configure the following **Optional Settings**:

    - **Tags**: Optionally, add tags that you can use to filter and group Sources in Cribl Edge's UI. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

4. Optionally, you can adjust the Authentication, TLS, Persistent Queue Settings, Processing, and Advanced settings, or Connected Destinations outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

Authentication

Select one of the following options for authentication:

- **None**: Don't use authentication.
- **Auth token**: Use HTTP token authentication. In the resulting **Token** field, enter the bearer token that must be included in the HTTP authorization header, or select **Generate** if you need a new token.
- **Auth token (text secret)**: Provide an HTTP token referenced by a secret. Select a stored text secret in the resulting drop-down, or select **Create** to configure a new secret.
- **Basic**: Displays **Username** and **Password** fields for you to enter HTTP Basic authentication credentials. Click **Generate** if you need a new password.
- **Basic (credentials secret)**: Provide username and password credentials referenced by a secret. Select a stored text secret in the resulting **Credentials secret** drop-down, or select **Create** to configure a new secret.

TLS Settings (Server Side)

Enabled: Defaults to toggled off. When toggled on:

Certificate: Name of the predefined certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

Passphrase: Passphrase to use to decrypt private key.

Certificate path: Server path containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

CA certificate path: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

Authenticate client (mutual auth): Require clients to present their certificates. Used to perform mutual authentication using SSL certs. Default is toggled off. When toggled on:

- **Validate client certificates**: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (for example, the system's CA). Default is toggled off.

- **Common name**: Regex that a peer certificate's subject attribute must match in order to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. (For example, to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.) If the subject attribute contains Subject Alternative Name (SAN) entries, the Source will check the regex against all of those but ignore the Common Name (CN) entry (if any). If the certificate has no SAN extension, the Source will check the regex against the single name in the CN.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In the **Persistent Queue Settings** tab, you can optionally specify persistent queue storage, using the following controls. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Source Persistent Queues (sPQ)

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the **Enable persistent queue** toggle. If enabled, PQ is automatically configured in `Always On` mode, with a maximum queue size of 1 GB disk space allocated per PQ-enabled Source, per Worker Process.

The 1 GB limit is on uncompressed inbound data, and the queue does not perform any compression. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable persistent queue: Default is toggled off. When toggled on:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to the Cribl Stream data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from the Cribl Stream data processing engine.

`Smart` mode only engages when necessary, such as when a downstream Destination becomes blocked *and* the **Buffer size limit** reaches its limit. When persistent queue is set to `Smart` mode, Cribl attempts

to flush the queue when every new event arrives. The only time events stay in the buffer is when a downstream Destination becomes blocked.

Buffer size limit: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. This buffer is for all connections, not just per Worker Process. For that reason, this can dramatically expand memory usage. Connections share this limit, which may result in slightly lower throughput for higher numbers of connections. For higher numbers of connections, consider increasing the limit.

Commit frequency: The number of events to send downstream before committing that Stream has read them. Defaults to `42`.

File size limit: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, and so forth. If not specified, Cribl Stream applies the default `1 MB`.

Queue size limit: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, and so forth. Can be set as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** in Group or Fleet settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file closes. Defaults to `None`; `Gzip` is also available.

In Cribl Stream 4.1 and newer, the Source persistent queue default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Optimize Source Persistent Queues (sPQ).

You can optimize Workers' startup connections and CPU load at **Group/Fleet settings** > Worker Processes.

Processing Settings Fields

In this section, you can define new fields or modify existing ones using JavaScript expressions, similar to the Eval function.

- The **Field Name** can either be a new field (unique within the event) or an existing field name to modify its value.
- The **Value** is a JavaScript expression (enclosed in quotes or backticks) to compute the field's value (can be a constant). Select this field's advanced mode icon (far right) if you'd like to open a modal where you can work with sample data and iterate on results.

This flexibility means you can:

- Add new fields to enrich the event.
- Modify existing fields by overwriting their values.
- Compute logic or transformations using JavaScript expressions.

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline or Pack to process data from this input before the data is sent through the Routes.

Advanced Settings

Show originating IP: Toggle on when clients are connecting through a proxy that supports the `X-Forwarded-For` header to keep the client's original IP address on the event instead of the proxy's IP address. This setting affects how the Source handles the `__srcIpPort` field.

Capture request headers: Toggle on to add request headers to events, in the `__headers` field.

Health check endpoint: Toggle on to enable a health check endpoint specific to this Source, `http(s)://<host>:<port>/cribl_health`. A `200` HTTP response code is returned when the Source is healthy. Otherwise, two errors you could receive are:

- `ECONNRESET` where the Source failed to initialize due to not having listeners on the port.
- `503` or `Server is busy, max active connections reached` indicate there are too many connections per Worker Process.

Active request limit (on-prem only): Maximum number of active requests allowed for this Source, per Worker Process. Defaults to `256`. Enter `0` for unlimited.



Raising this limit can increase throughput by allowing more concurrent data requests, but increases resource usage and load on both your Cribl Edge infrastructure and on downstream Destinations. Before raising this limit, ensure:

- Your Cribl Edge deployment has sufficient capacity to support higher request concurrency, including CPU, memory, or number of Worker Processes.
- Downstream Destinations are correctly sized and tuned to accept the higher data ingest rate, preventing backpressure. See Manage Backpressure for more information.

Improper sizing on either side can result in dropped events, delayed processing, or overall system instability.

Activity log sample rate: Determines how often request activity is logged at the `info` level. The default `100` value logs every 100th value; a `1` value would log every request; a `10` value would log every 10th request; etc.

Requests-per-socket limit: The maximum number of requests Cribl Edge should allow on one socket before instructing the client to close the connection. Defaults to `0` (unlimited). See Balancing Connection Reuse Against Request Distribution below.

Socket timeout (seconds): How long Cribl Edge should wait before assuming that an inactive socket has timed out. The default `0` value means wait forever.

Request timeout (seconds): How long to wait for an incoming request to complete before aborting it. The default `0` value means wait indefinitely.

Keep-alive timeout (seconds): After the last response is sent, Cribl Edge will wait this long for additional data before closing the socket connection. Defaults to `5` seconds; minimum is `1` second; maximum is `600` seconds (10 minutes).

> The longer the **Keep-alive timeout**, the more Cribl Edge will reuse connections. The shorter the timeout, the closer Cribl Edge gets to creating a new connection for every request. When request frequency is high, you can use longer timeouts to reduce the number of connections created, which mitigates the associated cost.

IP allowlist regex: Grants access to requests originating from specific IP addresses that match a defined pattern. Unmatched requests are rejected with a 403 (Forbidden) status code. Defaults to `.*` (allow all).

IP denylist regex: Blocks requests originating from specific IP addresses that match a defined pattern, even if they would be allowed by default. Rejected requests receive a 403 (Forbidden) status code. Defaults to `^$` (allow all).

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Balancing Connection Reuse Against Request Distribution

Requests-per-socket limit allows you to limit the number of HTTP requests an upstream client can send on one network connection. Once the limit is reached, Cribl Edge uses HTTP headers to inform the client that it must establish a new connection to send any more requests. (Specifically, Cribl Edge sets the HTTP `Connection` header to `close`.) After that, if the client disregards what Cribl Edge has asked it to do and tries to send another HTTP request over the existing connection, Cribl Edge will respond with an HTTP status code of `503 Service Unavailable`.

Use this setting to strike a balance between connection reuse by the client, and distribution of requests among one or more Edge Node processes by Cribl Edge:

- When a client sends a sequence of requests on the same connection, that is called connection reuse. Because connection reuse benefits client performance by avoiding the overhead of creating new connections, clients have an incentive to **maximize** connection reuse.
- Meanwhile, a single process on that Edge Node will handle all the requests of a single network connection, for the lifetime of the connection. When receiving a large overall set of data, Cribl Edge performs better when the workload is distributed across multiple Edge Node processes. In that situation, it makes sense to **limit** connection reuse.

There is no one-size-fits-all solution, because of variation in the size of the payload a client sends with a request and in the number of requests a client wants to send in one sequence. Start by estimating how long connections will stay open. To do this, multiply the typical time that requests take to process (based on payload size) times the number of requests the client typically wants to send.

If the result is 60 seconds or longer, set **Requests-per-socket limit** to force the client to create a new connection sooner. This way, more data can be spread over more Edge Node processes within a given unit of time.

For example: Suppose a client tries to send thousands of requests over a very few connections that stay open for hours on end. By setting a relatively low **Requests-per-socket limit**, you can ensure that the same work is done over more, shorter-lived connections distributed between more Edge Node processes, yielding better performance from Cribl Edge.

A final point to consider is that one Cribl Edge Source can receive requests from more than one client, making it more complicated to determine an optimal value for **Requests-per-socket limit**.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

Internal Fields

Cribl Edge uses a set of internal fields to assist in handling of data. These "meta" fields are **not** part of an event, but they are accessible, and Functions can use them to make processing decisions.

Fields for this Source:

- `__criblMetrics`
- `__final`
- `__headers` - Added only when **Advanced Settings** > **Capture request headers** is toggled on.

- `__inputId`
- `__srcIpPort` - See details below.
- `_time`
- `_value`

Overriding `__srcIpPort` with Client IP/Port

The `__srcIpPort` field's value contains the IP address and (optionally) port of the Prometheus Remote Write client sending data to this Source.

When any proxies (including load balancers) lie between the Prometheus Remote Write client and the Source, the last proxy adds an `X-Forwarded-For` header whose value is the IP/port of the original client. With multiple proxies, this header's value will be an array, whose first item is the original client IP/port.

If `X-Forwarded-For` is present, and **Advanced Settings** > **Show originating IP** is toggled off, the original client IP/port in this header will override the value of `__srcIpPort`.

If **Show originating IP** is toggled on, the `X-Forwarded-For` header's contents will **not** override the `__srcIpPort` value. (Here, the upstream proxy can convey the client IP/port without using this header.)

Detecting Metrics' Types

Because Prometheus remote write requests don't specify metrics' types, Cribl Edge applies the following rules to determine the type as we ingest them:

- If the metric's name ends with `_total`, `_sum`, `_count`, or `_bucket`, the type is set to `counter`.
- Otherwise, the metric's type is set to `gauge`.

This is consistent with the type detection practiced by other services implementing the remote write protocol. See, for example, New Relic's and Elastic's documentation.

Note that Cribl Edge supports the `timer` type in addition to `counter` and `gauge`.

Troubleshooting

The Source's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they are ingested. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the ingestion process, including any errors or warnings that may have occurred.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify ingestion issues. Analyze the graphs showing events and bytes in/out over time.

Common Issue

Dropping request because token invalid",“authToken”: “Bas...Njc=”

The specified token is invalid. Note that the above message is logged only at the debug level.

12.2.4.3. Grafana Source

Cribl Edge supports receiving metric and log data from Grafana Agent instances via the Prometheus remote write specification. The Grafana Agent uses Prometheus for metrics collection and Grafana Loki for log collection.

Type: **Push** | TLS Support: **YES** | Event Breaker Support: **No**

This Source assumes that incoming data is snappy-compressed.

Configure Cribl Edge to Receive Metrics and Logs from Grafana Agent Sources

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Source** and select the Source you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). Select the Source you want. Next, select **Add Source**.
2. In the **New Source** modal, configure the following under **General Settings**:
    - **Enabled**: Toggle on to enable the Source.
    - **Input ID**: Enter a unique name to identify this Source definition. If you clone this Source, Cribl Edge will add `-CLONE` to the original **Input ID**.
    - **Description**: Optionally, enter a description.
    - **Address**: Enter the hostname/IP to listen to. Defaults to `0.0.0.0`.
    - **Port**: Enter the port number to listen on.
3. Next, you can configure the following **Optional Settings**. However, at least one endpoint is required.
    - **Remote Write API endpoint**: Absolute path on which to listen for Grafana Agent's remote write requests. Defaults to `/api/prom/push`, which will (in this example) expand as: `http://<your-upstream-URL>:<your-port>/api/prom/push`.
    - **Logs API endpoint**: Absolute path on which to listen for Loki logs requests. Defaults to `/loki/api/v1/push`, which will (in this example) expand as: `http://<your-upstream-URL>:`

`<your-port>/loki/api/v1/push`.

- **Enable structured metadata**: Toggle on to parse and extract structured metadata from incoming Loki log payloads. Structured metadata lets you attach arbitrary key-value pairs (beyond labels) to each log entry, supporting high-cardinality fields without polluting the label set. Extracted metadata is stored in the event field `__structuredMetadata` to be forwarded through your Pipeline.
- **Tags**: Optionally, add tags that you can use to filter and group Sources in Cribl Edge's UI. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

4. Optionally, you can adjust the Authentication, TLS, Persistent Queue Settings, Processing, and Advanced settings, or Connected Destinations outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

Authentication

The **Authentication** tab provides separate **Loki** and **Prometheus** sections, enabling you to configure these inputs separately. The two sections provide identical options.

Select one of the following options for authentication:

- **None**: Don't use authentication.
- **Auth token**: Enter the bearer token that must be included in the authorization header.
- **Auth token (text secret)**: Provide an HTTP token referenced by a secret. Select a stored text secret in the resulting drop-down, or select **Create** to configure a new secret.
- **Basic**: Displays **Username** and **Password** fields for you to enter HTTP Basic authentication credentials.
- **Basic (credentials secret)**: Provide username and password credentials referenced by a secret. Select a stored text secret in the resulting **Credentials secret** drop-down, or select **Create** to configure a new secret.

TLS Settings (Server Side)

Enabled: Defaults to toggled off. When toggled on:

Certificate: Name of the predefined certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

Passphrase: Passphrase to use to decrypt private key.

Certificate path: Server path containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

CA certificate path: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

Authenticate client (mutual auth): Require clients to present their certificates. Used to perform mutual authentication using SSL certs. Default is toggled off. When toggled on:

- **Validate client certificates**: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (for example, the system's CA). Default is toggled off.
- **Common name**: Regex that a peer certificate's subject attribute must match in order to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. (For example, to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.) If the subject attribute contains Subject Alternative Name (SAN) entries, the Source will check the regex against all of those but ignore the Common Name (CN) entry (if any). If the certificate has no SAN extension, the Source will check the regex against the single name in the CN.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In the **Persistent Queue Settings** tab, you can optionally specify persistent queue storage, using the following controls. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Source Persistent Queues (sPQ)

> On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the **Enable persistent queue** toggle. If enabled, PQ is automatically configured in `Always On` mode, with a maximum queue size of 1 GB disk space allocated per PQ-enabled Source, per Worker Process.
>
> The 1 GB limit is on uncompressed inbound data, and the queue does not perform any compression. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable persistent queue: Default is toggled off. When toggled on:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to the Cribl Stream data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from the Cribl Stream data processing engine.



`Smart` mode only engages when necessary, such as when a downstream Destination becomes blocked *and* the **Buffer size limit** reaches its limit. When persistent queue is set to `Smart` mode, Cribl attempts to flush the queue when every new event arrives. The only time events stay in the buffer is when a downstream Destination becomes blocked.

Buffer size limit: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. This buffer is for all connections, not just per Worker Process. For that reason, this can dramatically expand memory usage. Connections share this limit, which may result in slightly lower throughput for higher numbers of connections. For higher numbers of connections, consider increasing the limit.

Commit frequency: The number of events to send downstream before committing that Stream has read them. Defaults to `42`.

File size limit: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, and so forth. If not specified, Cribl Stream applies the default `1 MB`.

Queue size limit: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data. Required, and defaults to `5 GB`. Accepts positive numbers with units of `KB`, `MB`, `GB`, and so forth. Can be set as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** in Group or Fleet settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file closes. Defaults to `None`; `Gzip` is also available.



In Cribl Stream 4.1 and newer, the Source persistent queue default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Optimize Source Persistent Queues (sPQ).

You can optimize Workers' startup connections and CPU load at **Group/Fleet settings** > Worker Processes.

Processing Settings Fields

In this section, you can define new fields or modify existing ones using JavaScript expressions, similar to the Eval function.

- The **Field Name** can either be a new field (unique within the event) or an existing field name to modify its value.
- The **Value** is a JavaScript expression (enclosed in quotes or backticks) to compute the field's value (can be a constant). Select this field's advanced mode icon (far right) if you'd like to open a modal where you can work with sample data and iterate on results.

This flexibility means you can:

- Add new fields to enrich the event.
- Modify existing fields by overwriting their values.
- Compute logic or transformations using JavaScript expressions.

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline or Pack to process data from this input before the data is sent through the Routes.

Advanced Settings

Show originating IP: Toggle on when clients are connecting through a proxy that supports the `X-Forwarded-For` header to keep the client's original IP address on the event instead of the proxy's IP address. This setting affects how the Source handles the `__srcIpPort` field.

Capture request headers: Toggle on to add request headers to events, in the `__headers` field.

Health check endpoint: Toggle on to enable a health check endpoint specific to this Source, `http(s)://<host>:<port>/cribl_health`. A `200` HTTP response code is returned when the Source is healthy. Otherwise, two errors you could receive are:

- `ECONNRESET` where the Source failed to initialize due to not having listeners on the port.
- `503` or `Server is busy, max active connections reached` indicate there are too many connections per Worker Process.

Active request limit (on-prem only): Maximum number of active requests allowed for this Source, per Worker Process. Defaults to `256`. Enter `0` for unlimited.



Raising this limit can increase throughput by allowing more concurrent data requests, but increases resource usage and load on both your Cribl Edge infrastructure and on downstream Destinations. Before raising this limit, ensure:

- Your Cribl Edge deployment has sufficient capacity to support higher request concurrency, including CPU, memory, or number of Worker Processes.
- Downstream Destinations are correctly sized and tuned to accept the higher data ingest rate, preventing backpressure. See Manage Backpressure for more information.

Improper sizing on either side can result in dropped events, delayed processing, or overall system instability.

Activity log sample rate: Determines how often request activity is logged at the `info` level. The default `100` value logs every 100th value; a `1` value would log every request; a `10` value would log every 10th request; and so forth.

Requests-per-socket limit: The maximum number of requests Cribl Edge should allow on one socket before instructing the client to close the connection. Defaults to `0` (unlimited). See Balancing Connection Reuse Against Request Distribution below.

Socket timeout (seconds): How long Cribl Edge should wait before assuming that an inactive socket has timed out. The default `0` value means wait forever.

Request timeout (seconds): How long to wait for an incoming request to complete before aborting it. The default `0` value means wait indefinitely.

Keep-alive timeout (seconds): After the last response is sent, Cribl Edge will wait this long for additional data before closing the socket connection. Defaults to `5` seconds; minimum is `1` second; maximum is `600` seconds (10 minutes).

The longer the **Keep-alive timeout**, the more Cribl Edge will reuse connections. The shorter the timeout, the closer Cribl Edge gets to creating a new connection for every request. When request frequency is high, you can use longer timeouts to reduce the number of connections created, which mitigates the associated cost.

IP allowlist regex: Grants access to requests originating from specific IP addresses that match a defined pattern. Unmatched requests are rejected with a 403 (Forbidden) status code. Defaults to `.*` (allow all).

IP denylist regex: Blocks requests originating from specific IP addresses that match a defined pattern, even if they would be allowed by default. Rejected requests receive a 403 (Forbidden) status code. Defaults to `^$` (allow all).

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Balancing Connection Reuse Against Request Distribution

Requests-per-socket limit allows you to limit the number of HTTP requests an upstream client can send on one network connection. Once the limit is reached, Cribl Edge uses HTTP headers to inform the client that it must establish a new connection to send any more requests. (Specifically, Cribl Edge sets the HTTP `Connection` header to `close`.) After that, if the client disregards what Cribl Edge has asked it to do and tries to send another HTTP request over the existing connection, Cribl Edge will respond with an HTTP status code of `503 Service Unavailable`.

Use this setting to strike a balance between connection reuse by the client, and distribution of requests among one or more Edge Node processes by Cribl Edge:

- When a client sends a sequence of requests on the same connection, that is called connection reuse. Because connection reuse benefits client performance by avoiding the overhead of creating new connections, clients have an incentive to **maximize** connection reuse.
- Meanwhile, a single process on that Edge Node will handle all the requests of a single network connection, for the lifetime of the connection. When receiving a large overall set of data, Cribl Edge performs better when the workload is distributed across multiple Edge Node processes. In that situation, it makes sense to **limit** connection reuse.

There is no one-size-fits-all solution, because of variation in the size of the payload a client sends with a request and in the number of requests a client wants to send in one sequence. Start by estimating how long connections will stay open. To do this, multiply the typical time that requests take to process (based on payload size) times the number of requests the client typically wants to send.

If the result is 60 seconds or longer, set **Requests-per-socket limit** to force the client to create a new connection sooner. This way, more data can be spread over more Edge Node processes within a given unit of time.

For example: Suppose a client tries to send thousands of requests over a very few connections that stay open for hours on end. By setting a relatively low **Requests-per-socket limit**, you can ensure that the same work is done over more, shorter-lived connections distributed between more Edge Node processes, yielding better performance from Cribl Edge.

A final point to consider is that one Cribl Edge Source can receive requests from more than one client, making it more complicated to determine an optimal value for **Requests-per-socket limit**.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

Internal Fields

Cribl Edge uses a set of internal fields to assist in handling of data. These "meta" fields are **not** part of an event, but they are accessible, and Functions can use them to make processing decisions.

Fields for this Source:

- `__headers` - Added only when **Advanced Settings** > **Capture request headers** is toggled on.
- `__inputId`
- `__labels` - For log events only - will contain all the labels found in each event's corresponding Loki stream.
- `__structuredMetadata` - Added only when **Enable structured metadata** is toggled on.
- `__srcIpPort` - See details below.

Overriding `__srcIpPort` with Client IP/Port

The `__srcIpPort` field's value contains the IP address and (optionally) port of the Grafana client sending data to this Source.

When any proxies (including load balancers) lie between the Grafana client and the Source, the last proxy adds an `X-Forwarded-For` header whose value is the IP/port of the original client. With multiple proxies, this header's value will be an array, whose first item is the original client IP/port.

If `X-Forwarded-For` is present, and **Advanced Settings** > **Show originating IP** is toggled off, the original client IP/port in this header will override the value of `__srcIpPort`.

If **Show originating IP** is toggled on, the `X-Forwarded-For` header's contents will **not** override the `__srcIpPort` value. (Here, the upstream proxy can convey the client IP/port without using this header.)

Detecting Metrics' Types

Because Prometheus remote write requests don't specify metrics' types, Cribl Edge applies the following rules to determine the type as we ingest them:

- If the metric's name ends with `_total`, `_sum`, `_count`, or `_bucket`, the type is set to `counter`.
- Otherwise, the metric's type is set to `gauge`.

This is consistent with the type detection practiced by other services implementing the remote write protocol. See, for example, New Relic's and Elastic's documentation.

Note that Cribl Edge supports the `timer` type in addition to `counter` and `gauge`.

Troubleshooting

The Source's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they are ingested. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the ingestion process, including any errors or warnings that may have occurred.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify ingestion issues. Analyze the graphs showing events and bytes in/out over time.

Common Issue

Dropping request because token invalid",“authToken”: “Bas...Njc=”

The specified token is invalid. Note that the above message is logged only at the debug level.

12.2.4.4. Loki Source

Cribl Edge supports receiving log data from Grafana Loki via an adaptation of the Protobuf (Protocol Buffers) specification.

Type: **Push** | TLS Support: **YES** | Event Breaker Support: **No**

This Source assumes that incoming data is snappy-compressed.

Configure Cribl Edge to Receive Loki Logs Data

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Source** and select the Source you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). Select the Source you want. Next, select **Add Source**.
2. In the **New Source** modal, configure the following under **General Settings**:
    - **Enabled**: Toggle on to enable the Source.
    - **Input ID**: Enter a unique name to identify this Source definition. If you clone this Source, Cribl Edge will add `-CLONE` to the original **Input ID**.
    - **Description**: Optionally, enter a description.
    - **Address**: Enter the hostname/IP to listen to. Defaults to `0.0.0.0`.
    - **Port**: Enter the port number to listen on.
    - **Logs API endpoint**: Absolute path on which to listen for Loki logs requests. Defaults to `/loki/api/v1/push`, which will (in this example) expand as: `http://<your-upstream-URL>:<your-port>/loki/api/v1/push`.
    - **Enable structured metadata**: Toggle on to parse and extract structured metadata from incoming Loki log payloads. Structured metadata lets you attach arbitrary key-value pairs (beyond labels) to each log entry, supporting high-cardinality fields without polluting the label set. Extracted metadata is stored in the event field `__structuredMetadata` to be forwarded through your Pipeline.

3. Next, you can configure the following **Optional Settings**: - **Tags**: Optionally, add tags that you can use to filter and group Sources in Cribl Edge's UI. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
4. Optionally, you can adjust the Authentication, TLS, Persistent Queue Settings, Processing, and Advanced settings, or Connected Destinations outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

Authentication

Use the **Authentication type** drop-down to specify how Loki's Promtail agent will authenticate against Cribl Edge:

- **None**: Don't use authentication.
- **Auth token**: Use HTTP token authentication. In the resulting **Token** field, enter the bearer token that must be included in the HTTP authorization header.
- **Auth token (text secret)**: Provide an HTTP token referenced by a secret. Select a stored text secret in the resulting drop-down, or select **Create** to configure a new secret.
- **Basic**: Displays **Username** and **Password** fields for you to enter HTTP Basic authentication credentials.
- **Basic (credentials secret)**: Provide username and password credentials referenced by a secret. Select a stored text secret in the resulting **Credentials secret** drop-down, or select **Create** to configure a new secret.

TLS Settings (Server Side)

Enabled: Defaults to toggled off. When toggled on:

Certificate: Name of the predefined certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

Passphrase: Passphrase to use to decrypt private key.

Certificate path: Server path containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

CA certificate path: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

Authenticate client (mutual auth): Require clients to present their certificates. Used to perform mutual authentication using SSL certs. Default is toggled off. When toggled on:

- **Validate server certs**: Toggle on to reject certificates that are **not** authorized by a CA in the **CA certificate path**, nor by another trusted CA (for example, the system's CA).

- **Common name**: Regex that a peer certificate's subject attribute must match in order to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. (For example, to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.) If the subject attribute contains Subject Alternative Name (SAN) entries, the Source will check the regex against all of those but ignore the Common Name (CN) entry (if any). If the certificate has no SAN extension, the Source will check the regex against the single name in the CN.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In the **Persistent Queue Settings** tab, you can optionally specify persistent queue storage, using the following controls. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Source Persistent Queues (sPQ)

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the **Enable persistent queue** toggle. If enabled, PQ is automatically configured in `Always On` mode, with a maximum queue size of 1 GB disk space allocated per PQ-enabled Source, per Worker Process.

The 1 GB limit is on uncompressed inbound data, and the queue does not perform any compression. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable persistent queue: Default is toggled off. When toggled on:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to the Cribl Stream data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from the Cribl Stream data processing engine.



`Smart` mode only engages when necessary, such as when a downstream Destination becomes blocked *and* the **Buffer size limit** reaches its limit. When persistent queue is set to `Smart` mode, Cribl attempts

to flush the queue when every new event arrives. The only time events stay in the buffer is when a downstream Destination becomes blocked.

Buffer size limit: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. This buffer is for all connections, not just per Worker Process. For that reason, this can dramatically expand memory usage. Connections share this limit, which may result in slightly lower throughput for higher numbers of connections. For higher numbers of connections, consider increasing the limit.

Commit frequency: The number of events to send downstream before committing that Stream has read them. Defaults to `42`.

File size limit: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, and so forth. If not specified, Cribl Stream applies the default `1 MB`.

Queue size limit: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, and so forth. Can be set as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** in Group or Fleet settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file closes. Defaults to `None`; `Gzip` is also available.

In Cribl Stream 4.1 and newer, the Source persistent queue default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Optimize Source Persistent Queues (sPQ).

You can optimize Workers' startup connections and CPU load at **Group/Fleet settings** > Worker Processes.

Processing Settings Fields

In this section, you can define new fields or modify existing ones using JavaScript expressions, similar to the Eval function.

- The **Field Name** can either be a new field (unique within the event) or an existing field name to modify its value.
- The **Value** is a JavaScript expression (enclosed in quotes or backticks) to compute the field's value (can be a constant). Select this field's advanced mode icon (far right) if you'd like to open a modal where you can work with sample data and iterate on results.

This flexibility means you can:

- Add new fields to enrich the event.
- Modify existing fields by overwriting their values.
- Compute logic or transformations using JavaScript expressions.

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline or Pack to process data from this input before the data is sent through the Routes.

Advanced Settings

Show originating IP: Toggle on when clients are connecting through a proxy that supports the `X-Forwarded-For` header to keep the client's original IP address on the event instead of the proxy's IP address. This setting affects how the Source handles the `__srcIpPort` field.

Capture request headers: Toggle on to add request headers to events, in the `__headers` field.

Health check endpoint: Toggle on to enable a health check endpoint specific to this Source, `http(s)://<host>:<port>/cribl_health`. A `200` HTTP response code is returned when the Source is healthy. Otherwise, two errors you could receive are:

- `ECONNRESET` where the Source failed to initialize due to not having listeners on the port.
- `503` or `Server is busy, max active connections reached` indicate there are too many connections per Worker Process.

Active request limit (on-prem only): Maximum number of active requests allowed for this Source, per Worker Process. Defaults to `256`. Enter `0` for unlimited.



Raising this limit can increase throughput by allowing more concurrent data requests, but increases resource usage and load on both your Cribl Edge infrastructure and on downstream Destinations. Before raising this limit, ensure:

- Your Cribl Edge deployment has sufficient capacity to support higher request concurrency, including CPU, memory, or number of Worker Processes.
- Downstream Destinations are correctly sized and tuned to accept the higher data ingest rate, preventing backpressure. See Manage Backpressure for more information.

Improper sizing on either side can result in dropped events, delayed processing, or overall system instability.

Activity log sample rate: Determines how often request activity is logged at the `info` level. The default `100` value logs every 100th value; a `1` value would log every request; a `10` value would log every 10th request; and so forth.

Requests-per-socket limit: The maximum number of requests Cribl Edge should allow on one socket before instructing the client to close the connection. Defaults to `0` (unlimited). See Balancing Connection Reuse Against Request Distribution below.

Socket timeout (seconds): How long Cribl Edge should wait before assuming that an inactive socket has timed out. The default `0` value means wait forever.

Request timeout (seconds): How long to wait for an incoming request to complete before aborting it. The default `0` value means wait indefinitely.

Keep-alive timeout (seconds): After the last response is sent, Cribl Edge will wait this long for additional data before closing the socket connection. Defaults to `5` seconds; minimum is `1` second; maximum is `600` seconds (10 minutes).

> The longer the **Keep-alive timeout**, the more Cribl Edge will reuse connections. The shorter the timeout, the closer Cribl Edge gets to creating a new connection for every request. When request frequency is high, you can use longer timeouts to reduce the number of connections created, which mitigates the associated cost.

IP allowlist regex: Grants access to requests originating from specific IP addresses that match a defined pattern. Unmatched requests are rejected with a 403 (Forbidden) status code. Defaults to `.*` (allow all).

IP denylist regex: Blocks requests originating from specific IP addresses that match a defined pattern, even if they would be allowed by default. Rejected requests receive a 403 (Forbidden) status code. Defaults to `^$` (allow all).

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Balancing Connection Reuse Against Request Distribution

Requests-per-socket limit allows you to limit the number of HTTP requests an upstream client can send on one network connection. Once the limit is reached, Cribl Edge uses HTTP headers to inform the client that it must establish a new connection to send any more requests. (Specifically, Cribl Edge sets the HTTP `Connection` header to `close`.) After that, if the client disregards what Cribl Edge has asked it to do and tries to send another HTTP request over the existing connection, Cribl Edge will respond with an HTTP status code of `503 Service Unavailable`.

Use this setting to strike a balance between connection reuse by the client, and distribution of requests among one or more Edge Node processes by Cribl Edge:

- When a client sends a sequence of requests on the same connection, that is called connection reuse. Because connection reuse benefits client performance by avoiding the overhead of creating new connections, clients have an incentive to **maximize** connection reuse.
- Meanwhile, a single process on that Edge Node will handle all the requests of a single network connection, for the lifetime of the connection. When receiving a large overall set of data, Cribl Edge performs better when the workload is distributed across multiple Edge Node processes. In that situation, it makes sense to **limit** connection reuse.

There is no one-size-fits-all solution, because of variation in the size of the payload a client sends with a request and in the number of requests a client wants to send in one sequence. Start by estimating how long connections will stay open. To do this, multiply the typical time that requests take to process (based on payload size) times the number of requests the client typically wants to send.

If the result is 60 seconds or longer, set **Requests-per-socket limit** to force the client to create a new connection sooner. This way, more data can be spread over more Edge Node processes within a given unit of time.

For example: Suppose a client tries to send thousands of requests over a very few connections that stay open for hours on end. By setting a relatively low **Requests-per-socket limit**, you can ensure that the same work is done over more, shorter-lived connections distributed between more Edge Node processes, yielding better performance from Cribl Edge.

A final point to consider is that one Cribl Edge Source can receive requests from more than one client, making it more complicated to determine an optimal value for **Requests-per-socket limit**.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

Internal Fields

Cribl Edge uses a set of internal fields to assist in handling of data. These "meta" fields are **not** part of an event, but they are accessible, and Functions can use them to make processing decisions.

Fields for this Source:

- `__final`

- `__headers` - Added only when **Advanced Settings** > **Capture request headers** is toggled on.
- `__inputId`
- `__labels` - Will contain all the labels found in each event's corresponding Loki stream.
- `__structuredMetadata` - Added only when **Enable structured metadata** is toggled on.
- `__srcIpPort` - See details below.
- `_time`

Overriding `__srcIpPort` with Client IP/Port

The `__srcIpPort` field's value contains the IP address and (optionally) port of the Loki client sending data to this Source.

When any proxies (including load balancers) lie between the Loki client and the Source, the last proxy adds an `X-Forwarded-For` header whose value is the IP/port of the original client. With multiple proxies, this header's value will be an array, whose first item is the original client IP/port.

If `X-Forwarded-For` is present, and **Advanced Settings** > **Show originating IP** is toggled off, the original client IP/port in this header will override the value of `__srcIpPort`.

If **Show originating IP** is toggled on, the `X-Forwarded-For` header's contents will **not** override the `__srcIpPort` value. (Here, the upstream proxy can convey the client IP/port without using this header.)

Troubleshooting

The Source's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they are ingested. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the ingestion process, including any errors or warnings that may have occurred.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify ingestion issues. Analyze the graphs showing events and bytes in/out over time.

Common Issue

Dropping request because token invalid",“authToken”: “Bas...Njc=”

The specified token is invalid. Note that the above message is logged only at the debug level.

12.2.5. Splunk

You can use the following Sources to receive data from Splunk services.

Source	Description
Splunk HEC	Receive data over HTTP/S using the Splunk HEC
Splunk TCP	Receive Splunk data from Universal or Heavy Forwarders

12.2.5.1. Splunk HEC Source

Cribl Edge supports receiving data over HTTP/S using the Splunk HEC (HTTP Event Collector).

Type: **Push** | TLS Support: **YES** | Event Breaker Support: **YES**

This Source supports gzip-compressed inbound data when the `Content-Encoding: gzip` connection header is set.

For examples of optimizing incoming Splunk events for downstream services, see our Splunk Stream to Cribl Stream, Splunk to Elasticsearch, and Splunk to Exabeam guides.

Also examine several Splunk Forwarder/UF Packs on the Cribl Packs Dispensary. These provide processing Pipelines to optimize Splunk Windows events and Universal Forwarder internal events. You can directly import these and adapt them to your needs.

The Splunk HEC Source is part of Cribl Edge's default configuration. If you delete it, Cribl Edge will re-create it when defaults are applied again (for example, when you create a new Fleet or deploy a configuration bundle). To prevent it from receiving data, disable it instead of deleting it.

Looking for the Splunk HEC Source in **Cribl Search**? See Ingest Splunk HEC Events into Cribl Search.

Configure Cribl Edge to Receive Data over Splunk HEC

Cribl Edge ships with a Splunk HEC Source preconfigured to listen on Port 8088. You can clone or directly modify this Source to further configure it, and then enable it.

In Cribl Edge, configure Splunk HEC:

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Source** and select the Source you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). Select the Source you want. Next, select **Add Source**.
2. Configure the following under **General Settings**:

- **Input ID**: Enter a unique name to identify this Splunk HEC Source definition. The default Source is prefilled with the value `in_splunk_hec`, which can't be changed via the UI. If you clone this Source, Cribl Edge will add `-CLONE` to the original **Input ID**.
- **Description**: Optionally, enter a description.
- **Address**: Enter the hostname/IP on which to listen for HTTP(S) data. Such as: `localhost` or `0.0.0.0`. Supports IPv4 and IPv6 addresses.
- **Port**: Enter the port number.
- **Splunk HEC endpoint**: Absolute path on which to listen for the Splunk HTTP Event Collector API requests. Defaults to `/services/collector`.
  - This single endpoint supports JSON events via `/event`, raw events via `/raw`, and Splunk S2S events via `/s2s`. See the Splunk REST API endpoints documentation and the following examples. The Source will automatically detect where to forward the request.

3. Next, you can configure the following **Optional Settings**:
   - **Allowed Indexes**: List the values allowed in the HEC event index field. Allows wildcards. Leave blank to skip validation.
   - **Enable Splunk HEC acknowledgements**: Whether to enable Splunk HEC acknowledgments. Default is toggled off. See Working with HEC Acks below to learn about context and limitations.
   - **Tags**: Optionally, add tags that you can use to filter and group Sources in Cribl Edge's UI. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
4. Optionally, you can adjust the TLS, Persistent Queue Settings, Processing, Auth tokens and Advanced settings, or Connected Destinations outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

Working with HEC Acks

Cribl Edge sends a `200` or similar HTTP response code when it receives an event.

Some senders also send **ack requests**, which differ from events. This type of request asks Cribl Edge to acknowledge delivery of an array of event IDs. (A sender might **require** HEC acks to be enabled. Cribl does not maintain a comprehensive list of senders that require acks - please refer to your sender's documentation.)

How Cribl Edge responds depends on **Optional Settings** > **Enable Splunk HEC acknowledgements**:

- When **Enable Splunk HEC acknowledgements** is toggled on, Cribl Edge will respond to an ack request with an acknowledgement affirming that Cribl Edge received each of the events whose IDs were listed in the array.
- When **Enable Splunk HEC acknowledgements** is toggled off, Cribl Edge will respond to an ack request with a `400` error and text saying that `ACK` is disabled.

It makes sense to turn on Splunk HEC acknowledgements for senders that keep TCP connections open while waiting for an ack, because this behavior can exhaust available file descriptors.

There is a caveat to how HEC acks work in Cribl Edge: The ack response simply cites whatever event IDs appeared in the ack request, regardless of whether or not those events were ever received or processed by Cribl Edge. In that sense, what Cribl Edge sends is a "fake ack:" unlike acknowledgements in some other systems, the ack in itself is meaningless. It mainly serves to spare a sender from keeping TCP connections open needlessly.

To determine whether Cribl Edge successfully ingested an event, pay no attention to acks. Instead, verify that the sender received a `200` or similar HTTP response from the Splunk HEC source.

TLS Settings (Server Side)



On Cribl.Cloud, you can enable TLS using Cribl-provided certficates if desired.

Enabled: Defaults to toggled off. When toggled on:

Certificate: Name of the predefined certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

Passphrase: Passphrase to use to decrypt private key.

Certificate path: Server path containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

CA certificate path: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

Authenticate client (mutual auth): Require clients to present their certificates. Used to perform mutual authentication using SSL certs. Default is toggled off. When toggled on:

- **Validate client certificates**: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (such as, the system's CA). Default is toggled off.
- **Common name**: Regex that a peer certificate's subject attribute must match in order to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. (For example, to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.) If the subject attribute contains Subject Alternative Name (SAN) entries, the Source will check the regex against all of those but ignore the Common Name (CN) entry (if any). If the certificate has no SAN extension, the Source will check the regex against the single name in the CN.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In the **Persistent Queue Settings** tab, you can optionally specify persistent queue storage, using the following controls. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Source Persistent Queues (sPQ)

> On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the **Enable persistent queue** toggle. If enabled, PQ is automatically configured in `Always On` mode, with a maximum queue size of 1 GB disk space allocated per PQ-enabled Source, per Worker Process.
>
> The 1 GB limit is on uncompressed inbound data, and the queue does not perform any compression. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable persistent queue: Default is toggled off. When toggled on:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to the Cribl Stream data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from the Cribl Stream data processing engine.

> `Smart` mode only engages when necessary, such as when a downstream Destination becomes blocked *and* the **Buffer size limit** reaches its limit. When persistent queue is set to `Smart` mode, Cribl attempts to flush the queue when every new event arrives. The only time events stay in the buffer is when a downstream Destination becomes blocked.

Buffer size limit: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. This buffer is for all connections, not just per Worker Process. For that reason, this can dramatically expand memory usage. Connections share this limit, which may result in slightly lower throughput for higher numbers of connections. For higher numbers of connections, consider increasing the limit.

Commit frequency: The number of events to send downstream before committing that Stream has read them. Defaults to `42`.

File size limit: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, and so forth. If not specified, Cribl Stream applies the default `1 MB`.

Queue size limit: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data. Required, and defaults to `5 GB`. Accepts positive numbers with units of `KB`, `MB`, `GB`, and so forth. Can be set as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** in Group or Fleet settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file closes. Defaults to `None`; `Gzip` is also available.



In Cribl Stream 4.1 and newer, the Source persistent queue default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Optimize Source Persistent Queues (sPQ).

You can optimize Workers' startup connections and CPU load at **Group/Fleet settings** > Worker Processes.

Processing Settings Event Breakers

Splunk HEC Source Event Breakers only function with the `/raw` endpoint. The `/event` and `/s2s` endpoints operate on the assumption that the data they receive is already properly formatted into individual events and therefore do not utilize event breakers.

- `/raw`: Event Breakers split incoming raw, unstructured data into distinct events before processing them through Pipelines.
- `/event`: Event Breakers are not applied since data is already segmented in JSON format.
- `/s2s`: Event Breakers are not applied since data is already structured in the Splunk-to-Splunk format.

Event Breaker rulesets: A list of event breaking rulesets that will be applied to the input data stream before the data is sent through the Routes. Defaults to `System Default Rule`.

Event Breaker buffer timeout: How long (in milliseconds) the Event Breaker will wait for new data to be sent to a specific channel, before flushing out the data stream, as-is, to the Routes. Minimum `10` ms, default `10000` (10 sec), maximum `43200000` (12 hours).

Fields

In this section, you can add Fields to each event using Eval-like functionality.

Name: Field name.

Value: JavaScript expression to determine field's value (can be a constant).



Fields specified on the **Fields** tab will normally override fields of the same name in events. But you can specify that fields in events should override these fields' values.

In particular, where incoming events have no `index` field, this Source adds one with the literal value `default`. You can override this value by using **Add Field** to specify an `index` field, and then setting its **Value** to an expression of the following form: `index == 'default' ? 'myIndex' : index`

Fields here are evaluated and applied **after** any fields specified in the Auth Tokens section.

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline or Pack to process data from this input before the data is sent through the Routes.

Auth Tokens

Provides the authentication token required to access this Source's data. Ensure the token is valid and has the necessary permissions for data retrieval.



Not providing a token allows **unauthorized access**.

If you exclusively use secret tokens for your Splunk HEC Source, be aware that Cribl Edge versions older than 4.8.2 will not recognize them. This can lead to unintended consequences, such as unsecured data ingestion. To ensure compatibility and security, configure both regular HEC tokens and secret tokens, especially in environments with mixed versions.

Authentication method: Choose between manually entering a **Token** or selecting a stored **Secret**. If you opt for manual entry, type the token directly or **Generate** a new one. For added security, use the secrets store to select a stored **Token (text secret)** or **Create** a new one.

Enable token: Controls the status of the specified authentication token. When toggled on, the specified auth token is active and can be used for client access. If toggled off, the token is disabled, causing it to be skipped and ignored during initialization. This means that any client attempting to use a disabled token will not be granted access.

Disabling a token can be helpful when you need to temporarily stop receiving data from a particular service or application. Disabling a token prevents the service or application from sending data to your endpoint without having to completely delete the token.

See also Periodic Logging for information on how auth tokens affect product logging.

Description: Optional (but recommended), the description for this token. Although this field is optional, the token description becomes the display name for the token. You can select this display name to expand or collapse the token details.

Allowed indexes: Optionally, specify which indexes are permissible for events ingested using this token. Supports multiple index values and wildcards.

Fields: Fields to add to events referencing this token. Each field is a **Name/Value** pair.



Fields specified on the **Auth Tokens** tab will normally override fields of the same name in events. But you can specify that fields in events should override these fields' values.

In particular, where incoming events have no `index` field, this Source adds one with the literal value `default`. You can override this value by using **Add Field** to specify an `index` field, and then setting its **Value** to an expression of the following form: `index == 'default' ? 'myIndex' : index`

Fields here are evaluated and applied **before** any fields specified in the Fields section.

Advanced Settings

Use Universal Forwarder time zone (S2S only): Toggle on (default) to have Event Breakers determine the time zone for events based on Universal Forwarder-provided metadata when the time zone can't be inferred from the raw event.

Drop control fields: Toggle on (default) to drop Splunk control fields such as `crcSalt` and `savedPort`. Toggle off if you want to retain these fields and store the data in the `__ctrlFields` internal field.

Extract metrics (S2S only): Toggle on if you're using Splunk's S2S protocol. When enabled, Cribl will identify and handle Splunk metric events, ensuring that metric events generated by Splunk are processed and forwarded as metrics.

Emit per-token request metrics: Toggle on to emit per-token `<prefix>.http.perToken` and summary `<prefix>.http.summary` request metrics. This enables Periodic Logging.

CORS allowed origins: If you need to enable cross-origin resource sharing with Splunk senders, use this field to specify the HTTP origins to which Cribl Edge should send `Access-Control-Allow-*` headers. You can enter multiple domains and use wildcards. This and its companion **CORS allowed headers** option should seldom be needed - see Working with CORS Headers below. Both options are available in Cribl Edge 4.1.2 and newer.

CORS allowed headers: List HTTP headers that Cribl Edge will send in a CORS preflight response to your configured (above) **CORS allowed origins** as `Access-Control-Allow-Headers`. Enter `*` to allow all headers.

Show originating IP: Toggle on when clients are connecting through a proxy that supports the `X-Forwarded-For` header to keep the client's original IP address on the event instead of the proxy's IP address. This setting affects how the Source handles the `__srcIpPort` field.

Capture request headers: Toggle on to add request headers to events, in the `__headers` field.

Active request limit (on-prem only): Maximum number of active requests allowed for this Source, per Worker Process. Defaults to `256`. Enter `0` for unlimited.



Raising this limit can increase throughput by allowing more concurrent data requests, but increases resource usage and load on both your Cribl Edge infrastructure and on downstream Destinations. Before raising this limit, ensure:

- Your Cribl Edge deployment has sufficient capacity to support higher request concurrency, including CPU, memory, or number of Worker Processes.
- Downstream Destinations are correctly sized and tuned to accept the higher data ingest rate, preventing backpressure. See Manage Backpressure for more information.

Improper sizing on either side can result in dropped events, delayed processing, or overall system instability.

Activity log sample rate: Determines how often request activity is logged at the `info` level. The default `100` value logs every 100th value; a `1` value would log every request; a `10` value would log every 10th request; and so forth.

Requests-per-socket limit: The maximum number of requests Cribl Edge should allow on one socket before instructing the client to close the connection. Defaults to `0` (unlimited). See Balancing Connection Reuse Against Request Distribution below.

Socket timeout (seconds): How long Cribl Edge should wait before assuming that an inactive socket has timed out. The default `0` value means wait forever.

Request timeout (seconds): How long to wait for an incoming request to complete before aborting it. The default `0` value means wait indefinitely.

Keep-alive timeout (seconds): After the last response is sent, Cribl Edge will wait this long for additional data before closing the socket connection. Defaults to `5` seconds; minimum is `1` second; maximum is `600` seconds (10 minutes).

The longer the **Keep-alive timeout**, the more Cribl Edge will reuse connections. The shorter the timeout, the closer Cribl Edge gets to creating a new connection for every request. When request frequency is high, you can use longer timeouts to reduce the number of connections created, which mitigates the associated cost.

IP allowlist regex: Grants access to requests originating from specific IP addresses that match a defined pattern. Unmatched requests are rejected with a 403 (Forbidden) status code. Defaults to `.*` (allow all).

IP denylist regex: Blocks requests originating from specific IP addresses that match a defined pattern, even if they would be allowed by default. Rejected requests receive a 403 (Forbidden) status code. Defaults to `^$` (allow all).

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Balancing Connection Reuse Against Request Distribution

Requests-per-socket limit allows you to limit the number of HTTP requests an upstream client can send on one network connection. Once the limit is reached, Cribl Edge uses HTTP headers to inform the client that it must establish a new connection to send any more requests. (Specifically, Cribl Edge sets the HTTP `Connection` header to `close`.) After that, if the client disregards what Cribl Edge has asked it to do and tries to send another HTTP request over the existing connection, Cribl Edge will respond with an HTTP status code of `503 Service Unavailable`.

Use this setting to strike a balance between connection reuse by the client, and distribution of requests among one or more Edge Node processes by Cribl Edge:

- When a client sends a sequence of requests on the same connection, that is called connection reuse. Because connection reuse benefits client performance by avoiding the overhead of creating new connections, clients have an incentive to **maximize** connection reuse.
- Meanwhile, a single process on that Edge Node will handle all the requests of a single network connection, for the lifetime of the connection. When receiving a large overall set of data, Cribl Edge performs better when the workload is distributed across multiple Edge Node processes. In that situation, it makes sense to **limit** connection reuse.

There is no one-size-fits-all solution, because of variation in the size of the payload a client sends with a request and in the number of requests a client wants to send in one sequence. Start by estimating how long connections will stay open. To do this, multiply the typical time that requests take to process (based on payload size) times the number of requests the client typically wants to send.

If the result is 60 seconds or longer, set **Requests-per-socket limit** to force the client to create a new connection sooner. This way, more data can be spread over more Edge Node processes within a given unit of time.

For example: Suppose a client tries to send thousands of requests over a very few connections that stay open for hours on end. By setting a relatively low **Requests-per-socket limit**, you can ensure that the same work is done over more, shorter-lived connections distributed between more Edge Node processes, yielding better performance from Cribl Edge.

A final point to consider is that one Cribl Edge Source can receive requests from more than one client, making it more complicated to determine an optimal value for **Requests-per-socket limit**.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

Internal Fields

Cribl Edge uses a set of internal fields to assist in handling of data. These "meta" fields are **not** part of an event, but they are accessible, and Functions can use them to make processing decisions.

Fields for this Source:

- `__headers` - Added only when **Advanced Settings** > **Capture request headers** is toggled on.
- `__hecToken`
- `__inputId`
- `__ctrlFields`
- `__s2sVersion` - value can be either `v3` or `v4`. This field is present only when the Source is receiving S2S-encoded payloads.
- `__srcIpPort` - See details below.
- `__TZ`

Universal Forwarder Time Zone

The `__TZ` field uses the universal forwarder time zone to mitigate cases where incoming events have timestamp strings but no time zone information. Event Breakers use the `__TZ` field to derive time zone information, enabling them to set the `_time` field correctly. See Using the UF Time Zone for additional information.

Overriding `__srcIpPort` with Client IP/Port

The `__srcIpPort` field's value contains the IP address and (optionally) port of the Splunk HEC client sending data to this Source.

When any proxies (including load balancers) lie between the Splunk HEC client and the Source, the last proxy adds an `X-Forwarded-For` header whose value is the IP/port of the original client. With multiple proxies, this header's value will be an array, whose first item is the original client IP/port.

If `X-Forwarded-For` is present, and **Advanced Settings** > **Show originating IP** is toggled off, the original client IP/port in this header will override the value of `__srcIpPort`.

If **Show originating IP** is toggled on, the `X-Forwarded-For` header's contents will **not** override the `__srcIpPort` value. (Here, the upstream proxy can convey the client IP/port without using this header.)

Working with CORS Headers

The **CORS allowed origins** and **CORS allowed headers** settings are needed **only** when you want JavaScript code running in a web browser to send events to the Splunk HEC Source. In this (uncommon) scenario, the code in question sends a preflight request that includes CORS (Cross-Origin Resource Sharing) headers, and the Splunk HEC Source includes CORS headers in its response.

If the settings (and thus the CORS headers in the response) are correct, the browser will allow the code in question to complete requests to the Splunk HEC Source - that is, allow the code to read the Source's responses. (These settings are optional because for some web applications it's sufficient to send requests **without** reading responses.)

The CORS header dance begins when the web browser sends a request with a header stating its origin. What happens next depends on how you've configured the CORS header settings, as explained in the tables below. (For these examples, we'll assume an origin of `https://mycoolwebapp:3001`.)

CORS allowed origins value	What Splunk HEC Source does	Does Browser then allow code to read response?
`*`	Send `Access-Control-Allow-Origin: *` header.	Yes
Multiple wildcard values	If origin value matches any wildcard, send `Access-Control-Allow-Origin: https://mycoolwebapp:3001` header.	Yes
Nothing (not configured)	Send neither an `Access-Control-Allow-Origin` nor an `Access-Control-Allow-Headers` header.	No

The web browser's request might also include an `Access-Control-Request-Headers` header that lists one or more headers that it wants to use in a subsequent `POST` request (that is, the "actual" as opposed to preflight request). The Splunk HEC Source can then respond with an `Access-Control-Allow-Headers` header **if and only if** it's also sending the `Access-Control-Allow-Origin` header.

CORS allowed headers value	What Splunk HEC Source does	Does Browser then allow code to read response?
*	Send `Access-Control-Allow-Headers` header with the same list the client sent.	Yes
Multiple wildcard values	If any headers from the client's list match wildcards, send `Access-Control-Allow-Headers` header listing the matching headers.	Yes
Nothing (not configured)	Do not send an `Access-Control-Allow-Headers` header.	Probably not

Format and Endpoint Examples

Splunk HEC to Cribl Edge

- Configure Cribl Edge to listen on port `8088` with an auth token of `myToken42`.
- Send a payload to your Cribl Edge receiver.

Note: Token specification can be either `Splunk <token>` or `<token>`.

Splunk HEC - JSON Event Examples | Splunk HEC - Raw Event Example | Splunk HEC - Health Event Example

```
curl -k "http://<myCriblHost>:8088/services/collector/event" \
-H "Authorization: myToken42" \
-d '{"event":"this is a sample event ", "host":"myHost", "source":"mySource", "fie


curl -k "http://<myCriblHost>:8088/services/collector" \
-H "Authorization: myToken42" \
-d '{"event":"this is a sample event ", "host":"myHost", "source":"mySource", "fie

# Multiple Events
curl -k "http://<myCriblHost>:8088/services/collector" \
-H "Authorization: myToken42" \
-d '{"event":"this is a sample event ", "host":"myHost", "source":"mySource", "fie

# Metrics Events
curl -k "http://<myCriblHost>:8088/services/collector/event" \
-H "Authorization: myToken42" \
-d '{"event":"metric", "host":"myHost", "fields":{"_value":3850,"metric_name":"ker

curl -k "http://<myCriblHost>:8088/services/collector/event" \
-H "Authorization: myToken42" \
-d '{"host":"myHost", "fields":{"_value":3850,"metric_name":"kernel.entropy_avail"

# Send the auth token as a query parameter, with no additional configuration
curl -k "http://<myCriblHost>:8088/services/collector/event?token=mToken42" \
-d '{"event":"this is a sample event ", "host":"myHost", "source":"mySource", "fie
```

Copy

Splunk HEC to Cribl.Cloud

- Navigate to Cribl.Cloud's Splunk HEC Source > **Auth Tokens** tab.
- Copy your token out of the **Token** field.
- From the command line, use `https`, your Cribl.Cloud portal's Ingress Address and port, and the token's value.

Splunk HEC > Cribl Cloud endpoint

```
curl -k "https://default.main.<organizationId>.cribl.cloud:8088/services/collector" \
-H "Authorization: <token>" \
-d '{"event": "Goats are better than ponies."}{"event": "Goats are better climbers."}
```

Copy

With a Cribl.Cloud Enterprise plan, generalize the above URL's `default.main` substring to `<groupName>.<workspaceName>` when sending to other Fleets and/or Workspaces.

Periodic Logging

Cribl Edge emits metrics about incoming requests when **Advanced Settings** > **Emit per-token request metrics** is enabled. These metrics can be routed to a Destination by enabling the Cribl Metrics Source. There are two categories of metrics: per-token metrics provide details for each configured token, while summary metrics aggregate data across all tokens. If no tokens are configured, only summary metrics will be emitted.

Per-token metric names will be prefixed by `<basePrefix>.http.perToken.`, where `<basePrefix>` is the configured base prefix for the Cribl Metrics Source. The available metric names are listed below, excluding the prefix. As these are counter-type metrics, they will not be emitted if the value is zero.

- `bytesReceived`
- `numRequests`
- `numEvents`
- `numErrors` *Note: this is cumulative of specific error types below as well as a catch-all for other types*
- `numParserErrors`
- `numToDisabledToken`

The summary metric names will be prefixed by `<basePrefix>.http.summary.` where `<basePrefix>` is the configured base prefix for the Cribl Metrics Source. The available metric names are listed below, excluding the prefix. As these are counter-type metrics, they will not be emitted if the value is zero.

- `bytesReceived`
- `numRequests`
- `numEvents`
- `numErrors` *Note: this is cumulative of specific error types below as well as a catch-all for other types*
- `numParserErrors`
- `numToDisabledToken`
- `numAuthFailures`

- `numToWrongUrl`
- `numAckReqs`
- `numReqsAcked`

Troubleshooting

The Source's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they are ingested. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the ingestion process, including any errors or warnings that may have occurred.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify ingestion issues. Analyze the graphs showing events and bytes in/out over time.

Common Issues

"Malformed HEC event"

The event is missing both `event` and `fields` fields.

"Invalid token"

Auth token(s) are defined, but the token specified in the HEC request doesn't match any defined tokens, therefore it's invalid.

"Invalid index"

The Splunk HEC Source's **Allowed Indexes** field is configured with specific indexes, but the client's HTTP request didn't specify any of them.

"{"text":"Server is busy","code":9,"invalid-event-number":0}"

"Server is busy" is the equivalent of a 503 HTTP response code. The most likely cause is the **Active request limit** setting in the HEC input's Advanced Settings is insufficient to service the number of simultaneous HEC requests. Increase the value and monitor your clients to see if the 503 response is eliminated.

This is not logged in Cribl Stream, but would be found in the response payload sent to your HEC client.

“Could not find parser for URL path” and “Dropping request because

12.2.5.2. Splunk TCP Source

Cribl Edge supports receiving Splunk data from Universal or Heavy Forwarders.

Type: **Push** | TLS Support: **YES** | Event Breaker Support: **YES**

For examples of optimizing incoming Splunk events for downstream services, see our Splunk Stream to Cribl Stream, Splunk to Elasticsearch, and Splunk to Exabeam guides.

Also examine several Splunk Forwarder/UF Packs on the Cribl Packs Dispensary. These provide processing Pipelines to optimize Splunk Windows events and Universal Forwarder internal events. You can directly import these and adapt them to your needs.

Looking for the Splunk TCP Source in **Cribl Search**? See Ingest Splunk Forwarder Data into Cribl Search.

Configure Cribl Edge to Receive Splunk TCP Data

Cribl Edge ships with a Splunk TCP Source preconfigured to listen on Port 9997. You can clone or directly modify this Source to further configure it, and then enable it.

In Cribl Edge, configure Splunk TCP:

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Source** and select the Source you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). Select the Source you want. Next, select **Add Source**.
2. Configure the following under **General Settings**:
    - **Input ID**: Enter a unique name. The default Source is prefilled with the value `in_splunk_tcp`, which can't be changed via the UI. If you clone this Source, Cribl Edge will add `-CLONE` to the original **Input ID**.
    - **Description**: Optionally, enter a description.
    - **Address**: Enter hostname/IP to listen for Splunk data. For example, `localhost` or `0.0.0.0`.

- **Port**: Enter port number.

3. Next, you can configure the following **Optional Settings**:
   - **Tags**: Optionally, add tags that you can use to filter and group Sources in Cribl Edge's UI. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
4. Optionally, you can adjust the TLS, Persistent Queue Settings, Processing, Auth tokens and Advanced settings, or Connected Destinations outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

TLS Settings (Server Side)



On Cribl.Cloud, you can enable TLS using Cribl-provided certficates if desired.

Enabled: Defaults to toggled off. When toggled on:

Certificate: Name of the predefined certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

Passphrase: Passphrase to use to decrypt private key.

Certificate path: Server path containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

CA certificate path: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

Authenticate client (mutual auth): Require clients to present their certificates. Used to perform mutual authentication using SSL certs. Default is toggled off. When toggled on:

- **Validate client certificates**: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (for example, the system's CA). Default is toggled on.
- **Common name**: Regex that a peer certificate's subject attribute must match in order to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. (For example, to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.) If the subject attribute contains Subject Alternative Name (SAN) entries, the Source will check the regex against all of those but ignore the Common Name (CN) entry (if any). If the certificate has no SAN extension, the Source will check the regex against the single name in the CN.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In the **Persistent Queue Settings** tab, you can optionally specify persistent queue storage, using the following controls. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Source Persistent Queues (sPQ)

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the **Enable persistent queue** toggle. If enabled, PQ is automatically configured in `Always On` mode, with a maximum queue size of 1 GB disk space allocated per PQ-enabled Source, per Worker Process.

The 1 GB limit is on uncompressed inbound data, and the queue does not perform any compression. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable persistent queue: Default is toggled off. When toggled on:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to the Cribl Stream data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from the Cribl Stream data processing engine.



`Smart` mode only engages when necessary, such as when a downstream Destination becomes blocked *and* the **Buffer size limit** reaches its limit. When persistent queue is set to `Smart` mode, Cribl attempts to flush the queue when every new event arrives. The only time events stay in the buffer is when a downstream Destination becomes blocked.

Buffer size limit: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. This buffer is for all connections, not just per Worker Process. For that reason, this can dramatically expand memory usage. Connections share this limit, which may result in slightly lower throughput for higher numbers of connections. For higher numbers of connections, consider increasing the limit.

Commit frequency: The number of events to send downstream before committing that Stream has read them. Defaults to `42`.

File size limit: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, and so forth. If not specified, Cribl Stream applies the default `1 MB`.

Queue size limit: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data. Required, and defaults to `5 GB`. Accepts positive numbers with units of `KB`, `MB`, `GB`, and so forth. Can be set as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** in Group or Fleet settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file closes. Defaults to `None`; `Gzip` is also available.



In Cribl Stream 4.1 and newer, the Source persistent queue default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Optimize Source Persistent Queues (sPQ).

You can optimize Workers' startup connections and CPU load at **Group/Fleet settings** > Worker Processes.

Processing Settings Event Breakers

Event Breakers are applied only to raw, unparsed data. Data originating from Splunk Heavy Forwarders or Universal Forwarders with sourcetypes configured for event breaking or indexed extractions bypasses Event Breakers, as it is already segmented into discrete events.

Event Breaker rulesets: A list of event breaking rulesets that will be applied to the input data stream before the data is sent through the Routes. Defaults to `System Default Rule`.

Event Breaker buffer timeout: How long (in milliseconds) the Event Breaker will wait for new data to be sent to a specific channel, before flushing out the data stream, as-is, to the Routes. Minimum `10` ms, default `10000` (10 sec), maximum `43200000` (12 hours).

Fields

In this section, you can define new fields or modify existing ones using JavaScript expressions, similar to the Eval function.

- The **Field Name** can either be a new field (unique within the event) or an existing field name to modify its value.

- The **Value** is a JavaScript expression (enclosed in quotes or backticks) to compute the field's value (can be a constant). Select this field's advanced mode icon (far right) if you'd like to open a modal where you can work with sample data and iterate on results.

This flexibility means you can:

- Add new fields to enrich the event.
- Modify existing fields by overwriting their values.
- Compute logic or transformations using JavaScript expressions.

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline or Pack to process data from this input before the data is sent through the Routes.

Auth Tokens

Add Token: Click to add authorization tokens. Each token's section provides the fields listed below. If no tokens are specified, unauthorized access **will be permitted**.

Token: Shared secrets to be provided by any Splunk forwarder (`Authorization: <token>`). Click **Generate** to create a new secret.

Description: Optional description for this token.

Advanced Settings

Enable proxy protocol: Toggle on if the connection is proxied by a device that supports Proxy Protocol v1 or v2.

IP allowlist regex: Regex matching IP addresses that are allowed to establish a connection. Defaults to `.*` (such as, all IPs).

Active connection limit: Maximum number of active connections allowed per Worker Process. Defaults to `1000`. Set a lower value if connection storms are causing the Source to hang. Set `0` for unlimited connections.

Socket idle timeout (seconds): The duration that Cribl Edge will wait for activity on an idle TCP socket before closing the connection. Disabled when set to `0`, the default.

Forced socket termination timeout (seconds): The extra time the server waits before forcibly closing a socket that has been idle (**TCP socket idle timeout**) or exceeded its maximum lifespan (**TCP socket max lifespan**) but has not yet properly closed. This prevents resource leaks caused by unresponsive clients or network issues. Configure based on network latency and client behavior. Default: `30` seconds. Set to `0` to disable.

Socket max lifespan (seconds): The duration that a socket is allowed to remain open, regardless of activity. This setting prevents resource exhaustion (such as TCP pinning) by limiting the lifespan of connections. Configure based on expected connection durations and resource availability. Disabled when set to `0`, the default.

Max S2S version: The highest version of the Splunk-to-Splunk protocol to expose during handshake. Defaults to `v4`; `v3` is also available.

Use Universal Forwarder time zone: Displayed (and enabled by default) only when **Max S2S version** is set to `v4`. Provides Event Breakers with a `__TZ` field, which derives events' time zone from UF-provided metadata. See Using the UF Time Zone and Configuring a Splunk Forwarder, below.

Drop control fields: Toggle on (default) to drop Splunk control fields such as `crcSalt` and `savedPort`. Toggle off if you want to retain these fields and store the data in the `__ctrlFields` internal field.

Extract metrics: Toggle on if you're using Splunk's S2S protocol. When enabled, Cribl will identify and handle Splunk metric events, ensuring that metric events generated by Splunk are processed and forwarded as metrics.

Compression: This setting controls how Cribl Edge reacts to incoming compressed Splunk-to-Splunk `v4` data. Select the **Compression** type from the drop-down list:

- `Automatic`: Cribl Edge will automatically decompress incoming `S2S v4` data.
- `Disabled`: Cribl Edge will reject compressed `S2S v4` connections, closing them immediately. This is the default setting for all Sources, both pre-existing (in versions 4.11 and older) and newly created.

This setting is not available when **Max S2S version** is set to `v3`.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

Using the UF Time Zone

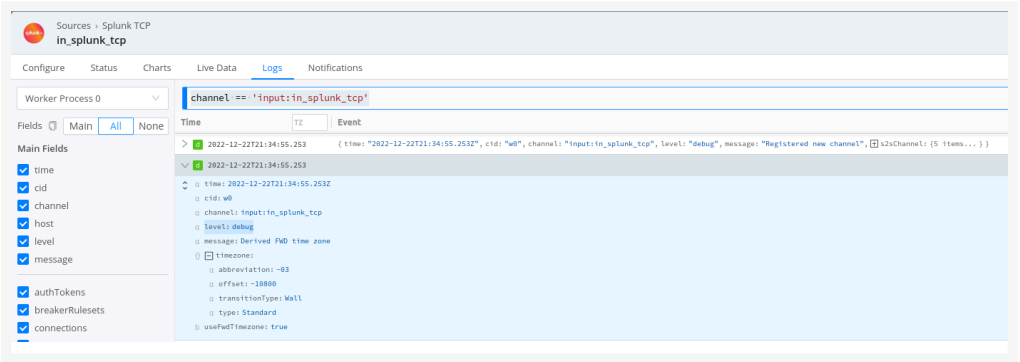
Under Advanced Settings, the **Use Universal Forwarder time zone** toggle mitigates cases where incoming events have timestamp strings but no time zone information. For example:

```
12-15-2022 14:57:22.080 WARN TcpOutputFd [1607 TcpOutEloop] - Connect to 172.17.0.1:9
```

Copy

This gap can be problematic, especially if the originating Universal Forwarder is in a different time zone from the processing Edge Node.

The `__TZ` field is the solution. Event Breakers use the `__TZ` field to derive time zone information, enabling them to set the `_time` field correctly. Derived time zone information will appear in Cribl Edge's own logs as shown below:



Time zone information in logs

Setting the Log Level for Connection Messages

When a Splunk forwarder connects to Cribl Edge, Cribl Edge logs the following message at the `debug` level: `Connection with forwarder has been established successfully.`

To see this message, set the `:forwarders` level to `debug`.

Each message contains details specific to the forwarder, such as the protocol, Splunk version, or remote host, to name a few.

In some situations, logging each incoming connection can produce many messages, which can make it hard to find other messages.

You can adjust the level of these connection messages. To do so, follow these steps:

1. Select **Settings** > **Global Settings** > **Logging** > **Levels**.
2. Search for the channel that logs the connection messages. It will have a name in the form `input:<source-id>:forwarders`. For example: `input:in_splunk_tcp:forwarders`.
3. Set the channel to the log level you prefer, such as `debug` or `silly`.
4. Click **Save** to save your setting.
5. Commit and deploy the change.

Internal Fields

Cribl Edge uses a set of internal fields to assist in handling of data. These "meta" fields are **not** part of an event, but they are accessible, and Functions can use them to make processing decisions.

Fields for this Source:

- `__inputId`
- `__ctrlFields`
- `__s2sVersion` - value can be either `v3` or `v4`
- `__source`
- `__srcIpPort`
- `__TZ` - see above

Configuring a Splunk Forwarder

To configure a Splunk forwarder (UF, HF) use the following sample outputs.conf stanzas:

outputs.conf (on-prem) | outputs.conf (Cribl Cloud)

```
[tcpout]
disabled = false
defaultGroup = cribl, <optional_clone_target_group>, ...
enableOldS2SProtocol = true

[tcpout:cribl]
server = [<cribl_ip>|<cribl_host>]:<port>, [<cribl_ip>|<cribl_host>]:<port>, ...
compressed = false
sendCookedData = true
# As of Splunk 6.5, using forceTimebasedAutoLB is no longer recommended. Ensure th
# forceTimebasedAutoLB = false
```

Copy

With a Cribl.Cloud Enterprise plan, generalize the above URL's `default.main` substring to `<groupName>.main` when sending to other Fleets.

If your use case requires compression, use useClientSSLCompression vs compressed on forwarder to compress the data stream.

Preventing Data Loss with v3

If you set **Max S2S version** to `v3` and are using Splunk 9.1.0 or later, Cribl recommends that you use the `enableOldS2SProtocol = true` setting shown above to avoid data loss. If you are working with `v3` and a Splunk version earlier than 9.1.0, you should use `negotiateProtocolLevel = 0`. Depending on your environment, enabling `negotiateProtocolLevel` with a non-`0` value could cause Cribl Edge to not accept data from the forwarder.

If you set **Max S2S version** to `v4`, these settings are not necessary. The Splunk receiver will detect which version is in use and automatically use the correct handler.

See Internal Fields for information on the `__s2sVersion` field.

Troubleshooting

The Source's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they are ingested. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the ingestion process, including any errors or warnings that may have occurred.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify ingestion issues. Analyze the graphs showing events and bytes in/out over time.

Splunk Forwarder Performance Issues

If you encounter performance issues with a Splunk Forwarder, Cribl recommends increasing the number of parallel ingestion Pipelines or increasing forwarder throughput. You can experiment with either or both of these settings.

To increase the number of parallel ingestion Pipelines, adjust the setting for `parallelIngestionPipelines` in `server.conf`. Experiment with values ranging from `2-4`.

To adjust forwarder throughput, increase the `maxKBps` value in `limits.conf`. The default value is `256`. A value of `0` removes all throttling from the forwarder.

"Connection rejected" with a `reason` of "Too many connections"

The maximum number of active Splunk TCP connections has been exceeded per Worker Process. The default is 1000.

Recommendation

In the Splunk TCP input's Advanced Settings configuration, increase the Max Active Connections value, set it to 0 for unlimited, and/or increase the # of Worker Processes the Worker Node(s) are using.

12.2.6. Windows

You can use the following Sources to receive data from Windows services.

Source	Description
Windows Event Forwarder	Receive events from Windows platforms
Windows Event Logs	Read events from the Windows Events API
Windows Metrics	Collect metrics from a Windows host, and populate dashboards

12.2.6.1. Windows Event Forwarder Source

Cribl Edge supports receiving Windows events from the Windows Event Forwarding (WEF) mechanism built into modern versions of Microsoft Windows (including Windows 10, Windows Server 2012, and more-recent releases).

The integration between WEF and Cribl works by generating events on Windows systems, which are collected by Windows Event Collectors (WECs). These events are then forwarded to Cribl using WEF, secured by either mutual TLS or Kerberos authentication. Cribl receives the events through the Windows Event Forwarder Source, processes them according to configured pipelines, and sends the processed events to designated Destinations.

Type: **Push** | TLS Support: **YES** | Event Breaker Support: **No** | Uses Key-Value Store

Looking for the Windows Event Forwarder Source in **Cribl Search**? See Ingest Windows Events into Cribl Search.

Prerequisites

- WEF is set up and enabled.
- WEF client(s) are pointed to WECs.
- You've configured either client certificate or Kerberos authentication.

On Cribl Edge, Kerberos authentication for this Source is supported only for Edge Nodes running on x86_64 Linux.

Configure Cribl Edge to Receive Windows Events

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Source** and select the Source you want from the list, choosing either **Select Existing**.
    - To configure via the Routes, select **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). Select the Source you want. Next, select **Add Source**.

2. In the Source modal, configure the following under **General Settings**:
    - **Input ID**: A unique name that identifies this Windows Event Forwarder Source.
    - **Address**: The hostname/IP on which to listen for Windows events data. Example: `localhost` or `0.0.0.0`.
    - **Port**: The port number. The default, `5986`, is the port used by Windows Event Collector for HTTPS-based subscriptions.
    - **Authentication method**: The method for accepting incoming client connections, either `Client certificate` or `Kerberos`.

Kerberos authentication is not supported for Cribl-managed Workers in Cribl.Cloud, but it is enabled for customer-managed Fleets, whether on-prem or hybrid.

    - For **Client Certificate**:
        - **Certificate name**: Choose the certificate you created.
        - **Private key path**: Server path containing the private key (in PEM format) to use. For Cribl.Cloud, the default path is `/opt/criblcerts/criblcloud.key` for a Cribl-managed Fleet in Cribl.Cloud. Path can reference `$ENV_VARS`.
        - **Passphrase**: Passphrase to use to decrypt private key.
        - **Certificate path**: Server path containing certificates (in PEM format) to use. For Cribl.Cloud, the default path is `/opt/criblcerts/criblcloud.crt` for a Cribl-managed Fleet in Cribl.Cloud. Path can reference `$ENV_VARS`.
        - **CA certificate path**: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`. If multiple certificates are present in a `.pem`, the first certificate in the chain should be the issuing CA certificate. If the first certificate is not the issuing CA, use the **CA fingerprint override** setting to specify the expected SHA1 fingerprint. This should match the order of traversing the chain from the host to the root CA cert.



The CA certificate that generated the Cribl Edge server certificate must match the root CA that signed all client certificates that will be forwarding Windows events to this Source. If the issuing CA is an intermediate CA, this CA must also match on the server and all client certs.

        - **Common name**: Regex that a peer certificate's subject attribute must be able to connect. Defaults to `.*`.
            - Matches on the substring after `CN=`.
            - As needed, escape regular expression (regex) tokens to match literal characters. For example, to match the subject `CN=worker.cribl.local`, you would enter:

`worker\.cribl\.local`.

- If the subject attribute contains Subject Alternative Name (SAN) entries, the Source will check the regex against all of those but ignore the Common Name (CN) entry (if any). If the certificate has no SAN extension, the Source will check the regex against the single name in the CN.

- Optionally, select the **Minimum TLS version** and **Maximum TLS version** to accept from connections.
- **Verify certificate via OCSP**: If toggled on, Cribl Edge will use an OCSP (Online Certificate Status Protocol) service to check that client certificates presented in incoming requests have not been revoked. Exposes the **Strict validation** toggle.
- **Strict validation**: If enabled, Cribl Edge will fail checks on **any** OCSP error. Otherwise, Cribl Edge will fail checks only when a certificate is revoked, and will ignore other errors (such as `OCSP server is unavailable` errors).

- For **Kerberos**: Provide the SPN and keytab file location that you set in Keytab file.
  - **Service Principal Name**: The Service Principal Name (SPN) in this format: `HTTP/<fully qualified domain name>@REALM`. This identifies the service in the Kerberos realm. The service pricipal name is case-sensitive, and must be exactly in the form used to export the keytab.
  - **Keytab location**: The path to the keytab file containing the service principal credentials. Cribl Edge uses `/etc/krb5.keytab` by default. This file contains the keys used for Kerberos authentication.
- Optionally, add **Tags** that you can use to filter and group Destinations in Cribl Edge's **Manage Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

3. **Subscriptions** define which logs to collect and how to filter them. You need to add at least one subscription. For comprehensive details on subscription settings and the query builder, refer to the subscriptions section.
4. Optionally, configure any Persistent Queue, Processing, and Advanced settings outlined in the below sections.
5. Click **Save**, then **Commit & Deploy**.

Subscriptions

Subscriptions control which events are collected from Windows endpoints and sent to the Cribl Edge Windows Event Forwarder Source. They filter the types of events to monitor, like security logs or application logs.



Due to the way subscription bookmarks are stored, Leader CPU usage is directly impacted by the number of subscriptions created for the Source and the number of clients receiving each subscription.

> To optimize performance, prefer to minimize the number of subscriptions by combining them where possible.

You can tailor your event collection to focus on the data that matters most to your Organization's security and monitoring needs. This helps reduce noise, save storage space, and ensure critical events are captured and analyzed.

Select **Add Subscription** and fill out the following fields to define at least one new, required subscription.

Subscription name: A friendly name for the subscription, which is only used to help you identify it.

Version: A read-only field that automatically updates whenever you save changes to the subscription. This version number reflects the specific configuration of the subscription and helps ensure that clients correctly interpret and process the subscription data.

Format: You can choose `Raw` to receive only XML data about the subscribed events, or `RenderedText` to additionally include amplifying information generated by the client about the event's contents.

Heartbeat: The maximum allowable time, in seconds, before the client will check in with Cribl Edge even if it has no new events to send.

Batch timeout: The maximum time, in seconds, that the client should batch new events before sending them to Cribl Edge.

> Windows Event Collector defines two delivery modes of "Event Delivery Optimization":
>
> - The "Minimize Bandwidth" mode corresponds to a **Heartbeat** and **Batch timeout** of 6 hours (21,600 seconds).
> - The "Minimize Latency" mode corresponds to a **Heartbeat** of 1 hour (3,600 seconds) and a **Batch timeout** of 30 seconds.

Locale: The language code to use for the events sent by the client, in RFC-3066 format (for example, `en-US`). The language code uses ISO 639, and the country code uses ISO 3166. Defaults to `en-US` when left blank.

Read existing events: Controls whether the client should send historical events when it first connects. The behavior of **Read existing events** is influenced by the **Use bookmarks** setting. See the table below for a complete overview.

Use bookmarks: Keeps track of which events it has received.

Setting	Behavior
Read existing events toggled off	Only new events generated after it receives the subscription are sent.

Setting	Behavior
Read existing events toggled on and **Use bookmarks** toggled off	All historical events are sent on the initial subscription and each time a client *forcibly* renews its subscription (for example, after group policy update), all events matching the query will be sent. Restarting Cribl Edge or restarting the client will not cause all events to be re-sent.
Read existing events toggled on and **Use bookmarks** toggled on	Historical events are sent on the initial subscription, but only new events (later than the saved bookmark) are sent on subsequent (even forced) subscription renewals.

Important Considerations

- **Subscription changes**: If you change and save a subscription, the saved bookmarks are no longer valid (they are tied to a specific subscription version), and all events matching the updated subscription will be sent.
- **Leader and Edge Node interactions**: Edge Nodes continue to accept new events even when the Leader is down. They sync with the Leader when it's back up. However, bookmarks stored in memory can be lost if a Edge Node restarts while the Leader is down.
- **Client balancing and bookmarks**: If a client balances to a different Edge Node while the Leader is down, it might receive a stale bookmark. Bookmarks are only sent to the client when it requests a subscription, not when it delivers events. This typically happens on the `Refresh=XX` interval from the GPO.
    - If the Leader is down when a client requests a subscription, it will receive the last known bookmark from the Edge Node, which could be outdated.
    - If a Edge Node restarts while the Leader is down, the client might not receive any bookmark at all, potentially leading to duplicate events.

Compression: Sets whether Windows clients compress the events sent to Cribl Edge, using the Streaming Lossless Data Compression (SLDC) algorithm. Defaults to toggled on.

Queries: See the explanation in Configuring Queries below.

Query builder mode: See the explanation in Configuring Queries below.

Targets: Set the DNS names of the clients that should receive this subscription. This field supports wildcard matching.

Fields: Use this setting to add fields exclusively to the events that WEF delivers **for the subscription you are defining**, using Eval-like functionality. To add fields to all incoming events regardless of subscription, use **Processing Settings** > **Fields** instead:

Optionally, you can create the field such that its **Value** is the value of a subscription metadata element. (See the available subscription metadata elements listed below.)

To do this, specify a **Value** of `__subscription.<metadata_element>`. The Source will then find the metadata element you specified in the special object `__subscription` and set the value to that.

For example, if you wanted all events for the subscription to be tagged with its **Locale**, you could set the **Name** to `Locale` (or any arbitrary name) and **Value** to `__subscription.locale`.

- **Field Name**: Unique name for the field you're adding.
- **Value**: JavaScript expression to compute field's value, enclosed in quotes or backticks. (Can evaluate to a constant.)

Subscription Metadata Elements

Metadata Element	Type	Corresponding UI Setting
`version`	string	**Version**
`subscriptionName`	string	**Name**
`contentFormat`	`Raw` or `RenderedText`	**Format**
`readExistingEvents`	boolean	**Read existing events**
`heartbeatInterval`	number	**Heartbeat**
`batchTimeout`	number	**Batch timeout**
`sendBookmarks`	boolean	**Use bookmarks**
`compress`	boolean	**Compression**
`queries`	array	see About the Query builder mode below

Change Subscriptions

If you change a subscription that is in use, Windows Event Forwarder will discover the new subscription ID. Once the `Refresh=XX` interval has elapsed, if any events were sent to the now-invalid previous subscription ID, Windows will resend them to the new one (assuming that these events remain valid under the updated subscription parameters). The Receiver Refuses to Accept Delivery of Events warnings you'll see in this scenario are normal.

About Query Builder Mode

This setting produces the `queries` array in one of two different forms, depending on whether you select its **Simple** button or its **Raw XML** button.

Simple produces an array whose elements each contain `path`, a string that corresponds to **Queries** > **Path**, and `queryExpression`, a string that corresponds to **Queries** > **Query expression**.

Raw XML produces an array whose elements each contain only the `xmlQuery` metadata element. This is the XML query you specified, in the form of a string, that corresponds to the **XML query** setting.

To access any element in the array, use array indexing. For example:

- `__subscription.queries[0].path` will access the path to the first query in an array created by **Query builder mode** > **Simple**.
- `__subscription.queries[0].xmlQuery` will access the first query in an array created by **Query builder mode** > **Raw XML**.

Configure Queries

Queries determine which events to send from the clients. **At least one query is required.** The format is derived from the XPath implementation used by Windows Event Collector.



While a single XPath query in a WEF subscription is limited to approximately 22 EventIDs, you can monitor more by creating a structured XML query with multiple `<Query>` elements within a single subscription. This is the recommended approach for performance and resource efficiency.

You have two **Query builder mode** options: **Simple** or **Raw XML**.

- **Raw XML** allows you to enter your XPath query into the **XML query** field.
- **Simple** allows you to manually build queries. Click **Add Query** to define each new query. A query has the following properties:
    - **Path**: Set this to the `Path` attribute of a `Select` XPath element. See the example below.
    - **Query expression**: Set this to the value inside a `Select` XPath element. See the example below.

With either a **Simple** or **Raw XML** query, you can use the **Targets** field to specify the DNS names of the endpoints that should forward these events. Supports wildcards (for example, `*.mydomain.com`). The default global wildcard (`*`) enables all endpoints.



If you have both **Simple** and **Raw XML** queries defined within a given Subscription, the **Query builder mode** button that you select when saving this Source's configuration determines which query/queries

Cribl Edge will send. To send both **Simple** and **Raw XML** queries, use the **Add Subscription** button to define multiple subscriptions.

Example

When creating a subscription in a Windows Event Collector, you can view the generated XML/XPath query. Consider a subscription that returns all events from the Security log that are of severities Critical, Error, or Warning, and that occurred within the last 24 hours. This subscription would generate XML like this:

```
<QueryList>
  <Query Id="0" Path="Security">
    <Select Path="Security">*[System[(Level=1  or Level=2 or Level=3) and TimeCreated
  </Query>
</QueryList>
```

Copy

To use this subscription in a Cribl Edge Windows Event Forwarder Source, you would either use the **Raw XML** option and paste the above XML, or use **Simple** mode and set the following query properties:

- **Path**: `Security`
- **Query expression**: `*[System[(Level=1 or Level=2 or Level=3) and TimeCreated[timediff(@SystemTime) <= 86400000]]]`

You do not need to use the `<Query>` element's `Id` or `Path` attributes anywhere in the Cribl Edge subscription config.

Persistent Queue Settings

In the **Persistent Queue Settings** tab, you can optionally specify persistent queue storage, using the following controls. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Source Persistent Queues (sPQ)

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the **Enable persistent queue** toggle. If enabled, PQ is automatically configured in `Always On` mode, with a maximum queue size of 1 GB disk space allocated per PQ-enabled Source, per Worker Process.

The 1 GB limit is on uncompressed inbound data, and the queue does not perform any compression. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable persistent queue: Default is toggled off. When toggled on:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to the Cribl Stream data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from the Cribl Stream data processing engine.

`Smart` mode only engages when necessary, such as when a downstream Destination becomes blocked *and* the **Buffer size limit** reaches its limit. When persistent queue is set to `Smart` mode, Cribl attempts to flush the queue when every new event arrives. The only time events stay in the buffer is when a downstream Destination becomes blocked.

Buffer size limit: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. This buffer is for all connections, not just per Worker Process. For that reason, this can dramatically expand memory usage. Connections share this limit, which may result in slightly lower throughput for higher numbers of connections. For higher numbers of connections, consider increasing the limit.

Commit frequency: The number of events to send downstream before committing that Stream has read them. Defaults to `42`.

File size limit: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, and so forth. If not specified, Cribl Stream applies the default `1 MB`.

Queue size limit: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data. Required, and defaults to `5 GB`. Accepts positive numbers with units of `KB`, `MB`, `GB`, and so forth. Can be set as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** in Group or Fleet settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file closes. Defaults to `None`; `Gzip` is also available.



In Cribl Stream 4.1 and newer, the Source persistent queue default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Optimize Source Persistent Queues (sPQ).

You can optimize Workers' startup connections and CPU load at **Group/Fleet settings** > Worker Processes.

Processing Settings Fields

Use this setting to add fields to **all events coming into the Source**, using Eval-like functionality. (To add fields on a per-subscription basis, use **Subscriptions** > **Fields** instead.)

Field Name: Unique name for the field you're adding.

Value: JavaScript expression to compute field's value, enclosed in quotes or backticks. (Can evaluate to a constant.)

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline or Pack to process data from this input before the data is sent through the Routes.

Advanced Settings

Allow MachineID mismatch (client certificate only): Toggle on if you do not want to verify that the events sent by a client match the client's certificate Common Name (Subject CN). If toggled off, events where the `MachineID` (by default the client's machine name, like `CLIENT1.domainName.com`) do not match the CN will be rejected.

This setting applies only to client certificate authentication. If you use Kerberos authentication, this option will not appear and will be treated as if it is toggled off.

Show originating IP: Toggle on when clients are connecting through a proxy that supports the `X-Forwarded-For` header to keep the client's original IP address on the event instead of the proxy's IP address. This setting affects how the Source handles the `__srcIpPort` field.

Capture request headers: Toggle on to add request headers to events, in the `__headers` field.

Health check endpoint: Toggle on to enable a health check endpoint specific to this Source: `http(s)://<host>:<port>/cribl_health`. A `200` HTTP response code is returned when the Source is healthy. Otherwise, two errors you could receive are:

- `ECONNRESET` where the Source failed to initialize due to not having listeners on the port.
- `503` or `Server is busy, max active connections reached` indicate there are too many connections per Worker Process.

Log CA fingerprint mismatch warning (client certificate only): Toggle on to log a warning message when the client certificate's issuing Certificate Authority (CA) fingerprint doesn't match the expected CA fingerprint configured in the Windows Event Forwarding Source. This mismatch prevents proper authentication and event forwarding. The log message is `Mismatched fingerprint` and includes both the expected `acceptedFingerprint` and actual `issuerFingerprint` fingerprints. Toggled off by default.

Active request limit (on-prem only): Maximum number of active requests allowed for this Source, per Worker Process. Defaults to `256`. Enter `0` for unlimited.



Raising this limit can increase throughput by allowing more concurrent data requests, but increases resource usage and load on both your Cribl Edge infrastructure and on downstream Destinations. Before raising this limit, ensure:

- Your Cribl Edge deployment has sufficient capacity to support higher request concurrency, including CPU, memory, or number of Worker Processes.
- Downstream Destinations are correctly sized and tuned to accept the higher data ingest rate, preventing backpressure. See Manage Backpressure for more information.

Improper sizing on either side can result in dropped events, delayed processing, or overall system instability.

Requests-per-socket limit: The maximum number of requests Cribl Edge should allow on one socket before instructing the client to close the connection. Defaults to `0` (unlimited). See Balancing Connection Reuse Against Request Distribution below.

Socket timeout (seconds): How long Cribl Edge should wait before assuming that an inactive socket has timed out. The default `0` value means wait forever. For Kerberos authentication, see how to adjust timeouts.

Keep-alive timeout (seconds): After the last response is sent, Cribl Edge how long to wait for additional data before closing the socket connection. Defaults to `90` seconds; minimum is `1` second; maximum is `600` seconds (10 minutes).

The longer the timeout, the more Cribl Edge will reuse connections. The shorter the timeout, the closer Cribl Edge gets to creating a new connection for every request. When request frequency is high, you can use longer timeouts to reduce the number of connections created, which mitigates the associated cost.

For Kerberos authentication, see how to adjust timeouts.

IP allowlist regex: Grants access to requests originating from specific IP addresses that match a defined pattern. Unmatched requests are rejected with a 403 (Forbidden) status code. Defaults to `.*` (allow all).

IP denylist regex: Blocks requests originating from specific IP addresses that match a defined pattern, even if they would be allowed by default. Rejected requests receive a 403 (Forbidden) status code. Defaults to `^$` (allow all).

CA fingerprint override (client certificate only): Use this setting only if the first certificate in the configured CA chain does not match the SHA1 fingerprint that the WEF client expects, such as an intermediate certificate. If that is the case, enter the expected SHA1 fingerprint. The thumbprint specified here must match the `IssuerCA` parameter in the Subscription Manager template.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Balancing Connection Reuse Against Request Distribution

Requests-per-socket limit allows you to limit the number of HTTP requests an upstream client can send on one network connection. Once the limit is reached, Cribl Edge uses HTTP headers to inform the client that it must establish a new connection to send any more requests. (Specifically, Cribl Edge sets the HTTP `Connection` header to `close`.) After that, if the client disregards what Cribl Edge has asked it to do and tries to send another HTTP request over the existing connection, Cribl Edge will respond with an HTTP status code of `503 Service Unavailable`.

Use this setting to strike a balance between connection reuse by the client, and distribution of requests among one or more Edge Node processes by Cribl Edge:

- When a client sends a sequence of requests on the same connection, that is called connection reuse. Because connection reuse benefits client performance by avoiding the overhead of creating new connections, clients have an incentive to **maximize** connection reuse.
- Meanwhile, a single process on that Edge Node will handle all the requests of a single network connection, for the lifetime of the connection. When receiving a large overall set of data, Cribl Edge performs better when the workload is distributed across multiple Edge Node processes. In that situation, it makes sense to **limit** connection reuse.

There is no one-size-fits-all solution, because of variation in the size of the payload a client sends with a request and in the number of requests a client wants to send in one sequence. Start by estimating how long connections will stay open. To do this, multiply the typical time that requests take to process (based on payload size) times the number of requests the client typically wants to send.

If the result is 60 seconds or longer, set **Requests-per-socket limit** to force the client to create a new connection sooner. This way, more data can be spread over more Edge Node processes within a given unit of time.

For example: Suppose a client tries to send thousands of requests over a very few connections that stay open for hours on end. By setting a relatively low **Requests-per-socket limit**, you can ensure that the same work is done over more, shorter-lived connections distributed between more Edge Node processes, yielding better performance from Cribl Edge.

A final point to consider is that one Cribl Edge Source can receive requests from more than one client, making it more complicated to determine an optimal value for **Requests-per-socket limit**.

Timeout Considerations for Kerberos Authentication

If you use Kerberos authentication, there are some trade-offs you should consider when you select settings in **Subscriptions** > **Batch timeout** and **Advanced Settings** > **Keep-alive timeout (seconds)**.

Windows expects socket connections to be long-lived for Kerberos authentication so it can avoid the overhead of performing the two-step Kerberos authentication on each new batch delivery.

Set **Subscriptions** > **Batch timeout**, **Subscriptions** > **Heartbeat**, or both to be less than or equal to **Keep-alive timeout** (or the lowest **Batch timeout** if several subscriptions are used), especially when a load balancer is in front of the Edge Nodes.

Set **Socket timeout** higher than **Batch timeout** (or the lowest **Batch timeout** if several subscriptions are used). This ensures that connections aren't prematurely closed before batches are sent. If the **Socket timeout** has elapsed between event deliveries or heartbeats, Cribl Edge will still handle it gracefully, but incur an additional overhead of two network round trips.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

Event Handling

The Windows Event Forwarder Source uses internal fields and a key-value store to manage and process events efficiently. Internal fields are metadata added to events for internal processing and should be treated as read-only to avoid unintended consequences. Key fields include:

- `__final`
- `__headers` - Added only when **Advanced Settings** > **Capture request headers** is toggled on.
- `__inputId`
- `__srcIpPort` - See details below.

- `__subscriptionName`
- `__subscriptionVersion`
- `_raw`
- `_time`

The key-value store uses the `machineId` as keys to store bookmark information when bookmarks are enabled. This ensures that events are processed correctly and efficiently by maintaining the state of event processing.

Together, the key-value store and internal fields work to ensure accurate and efficient event handling. The key-value store maintains the state of event processing, while internal fields provide necessary metadata for internal operations, enabling Cribl Stream to process and route events effectively.

Overriding `__srcIpPort` with Client IP/Port

The `__srcIpPort` field's value contains the IP address and (optionally) port of the WEF client sending data to this Source.

In the WEF Source configuration, there's an option to handle the `X-Forwarded-For` header, which is commonly used by load balancers to pass the original client's IP address. When any proxies (including load balancers) lie between the WEF client and the Source, the last proxy adds an `X-Forwarded-For` header whose value is the IP/port of the original client. With multiple proxies, this header's value will be an array, whose first item is the original client IP/port.

If `X-Forwarded-For` is present, and **Advanced Settings** > **Show originating IP** is toggled off, the original client IP/port in this header will override the value of `__srcIpPort`.

If **Show originating IP** is toggled on, the `X-Forwarded-For` header's contents will **not** override the `__srcIpPort` value. (Here, the upstream proxy can convey the client IP/port without using this header.)

Troubleshoot

The Source's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they are ingested. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the ingestion process, including any errors or warnings that may have occurred.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify ingestion issues. Analyze the graphs showing events and bytes in/out over time.

Common Issues High CPU Usage on Leader Node API Process

Problem: Experiencing high CPU usage on your Leader Node's API Process is often a symptom of having too many subscriptions (thousands). This can lead to performance issues and instability.

Solutions: First attempt to reduce or consolidate your subscriptions. This can often alleviate the problem without the need for additional configurations. However, since the process of consolidating subscriptions can vary significantly between environments, there is no one-size-fits-all approach.

If you are unable to reduce the number of subscriptions sufficiently, consider enabling Write-Ahead Log (WAL) mode for the key-value store (KV Store). We recommend working with Cribl Support to incorporate the new **KV Store** settings effectively. You can find these settings under **Settings** > **Global Settings** > **General Settings** > **Limits** or in the limits.yml file.

Security Event Log Forwarding Fails

Error: `Error "0x138C" Windows Event Forward plugin can't read any event from the query since the query returns no active channel.`

Solution: Enable the Network Service to read the security logs by configuring access for the Event Log Service.

When configuring a Group Policy Object (GPO) to add NETWORK SERVICE permissions, carefully review existing permissions to prevent unintended removal. Preserve all necessary permissions to avoid application issues.

Cannot Find the Certificate

Error: `The forwarder is having a problem communicating with subscription manager at address... The WS-Management service cannot find the certificate that was requested.`

For example:

```
The forwarder is having a problem communicating with subscription manager at address

Error code is 2150858882 and Error Message is <f:WSManFault xmlns:f=http://schemas.mi
```

Copy

Resolution:

- The `NETWORK SERVICE` user does not have permissions to use the private key of the certificate for authentication.
- Add the `NETWORK SERVICE` user to the list of users allowed to use the private key.

Receiver Refuses to Accept Delivery of Events

Warning: `The receiver refuses to accept delivery of events and requests that the subscription be canceled.`

If you make a change to a subscription that is in use, **and** if the Windows client tries to send events before the `Refresh=XX` interval has elapsed, the result is that these events are sent to the old subscription, which is now invalid because its query is outdated or because other subscription parameters have changed.

The warning message tells the WEF client to stop sending events to the now-invalid old subscription.

Resolution:

Windows Event Forwarder handles this scenario gracefully, and you do not need to take any action to resolve it:

- Once the `Refresh=XX` interval has elapsed, the WEF client will ask for and receive the updated subscription, and from then on should be able to successfully deliver events against that new subscription.
- Any events that were refused under the old/invalid subscription ID will be resent under the new one, assuming that they remain valid according to the new subscription parameters.

Subscription Cannot be Created

Error: `The subscription <subscription> can not be created. The error code is 5004.`

The error appears in the `Microsoft-Windows-Eventlog-ForwardingPlugin/Operational` log.

Resolution:

1. Apply the `O:BAG:SYD:...` permissions to make the log file readable.
2. Reboot the machine.

Verify that Client and Destination are Joined to a Domain

Error: `If Kerberos mechanism is used, verify that the client computer and the destination computer are joined to a domain.`

When you are using auto-enroll, the `Eventlog-ForwardingPlugin Operational` logs might display an error of this form.

Resolution:

In your auto-enroll template's **Client-Server Authentication Properties**, make sure the **Subject Name format** is set to **Common name**.

Resources

For processing Pipelines that you can import and adapt to your needs, examine these resources on the Cribl Packs Dispensary:

- Microsoft Windows Events - Shape events into JSON or Key=Value format, and dramatically reduce event sizes.
- Cribl Edge Windows for Common Schema - Transform Windows events into a Common Schema.
- Exabeam Pack for Windows - Forward specific Windows Event IDs to Exabeam.
- QRadar Pack for Windows Events - Optimize Windows events for delivery to QRadar.

Further Reading

12.2.6.1.1. Client Certificate Authentication For Windows Event Forwarder

This document covers the client certificate authentication setup steps you need to take before configuring a Windows Event Forwarder Source.

1. Set Up WEF

Follow this Microsoft guide for setting up Windows Event Forwarding (WEF) in a traditional Windows environment. Generally, you can follow the guide's "non-domain" section to correctly configure the endpoints/senders.

2. Configure Certificates

After setting up WEF, you need to configure certificates to ensure secure communication. Server certificate configuration varies depending on your deployment type. However, you'll always need to configure a client certificate.

On-prem deployments: Add the server certificate and CA chain to the Fleet, see importing certificates for details. Add the client certificate to Windows either manually or with certificate auto-enrollment through Group Policy to automatically configure unique device certificates. For specific configuration options, consult with your Windows administrators.

Cribl.Cloud deployment: Upload the CA chain for your client certificate, see importing certificates for details. The actual certificate/key pair in the chain doesn't matter for this purpose. It's solely to satisfy the technical requirement of uploading a chain. Use Cribl's provided certificate and key:

- `/opt/criblcerts/criblcloud.crt`
- `/opt/criblcerts/criblcloud.key`

3. Set Client Certificate Permissions

Set the appropriate permissions for the `NETWORK SERVICE` user to use the private key of the certificate for authentication.

Open the Certificate Manager tool (`certlm.msc`).

- Right-click your client certificate and select **All Tasks** > **Manage Private Keys...**.
- Add the `NETWORK SERVICE` user.

4. Configure the Subscription Manager and Log Forwarding Group Policies

1. On a domain controller, open the Group Policy Management Console (GPMC).
2. Create a new Group Policy Object (GPO) or edit an existing one that is linked to the organizational unit (OU) containing the computers you want to configure.
3. Navigate to **Computer Configuration** > **Policies** > **Administrative Templates** > **Windows Components**. Select **Event Forwarding** to open the Local Group Policy Editor.
4. Select **Configure target Subscription Manager** and use one of the following templates depending on your deployment type, substituting the appropriate values for the placeholders.
   On-Prem:

   ```
   Server=https://<cribl-worker-or-lb>:<wef-source-port>/wsman/SubscriptionManager/
   ```

   Copy

   Cribl.Cloud:

   ```
   https://<groupName>.main.<org-name>.cribl.cloud:<wef-source-port>/wsman/Subscrip
   ```

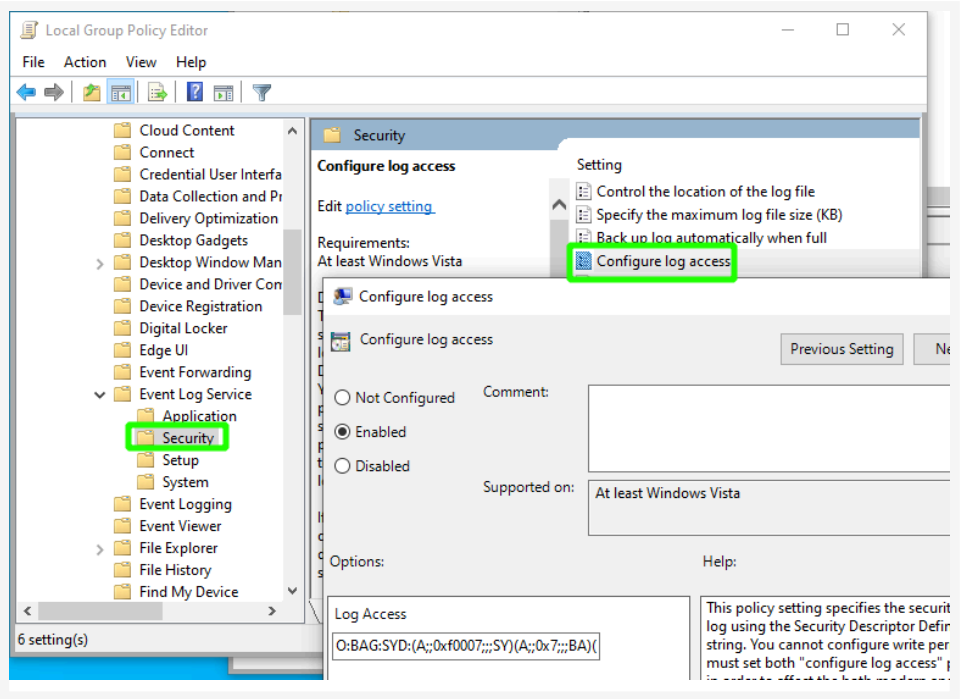
   Copy

   - The path portion `/wsman/SubscriptionManager/WEC` within the **Subscription Target URL** set using Group Policy Objects (GPOs) is the same as required for a WEC subscription and is strictly case-sensitive. If the capitalization is incorrect, WEF will encounter errors and fail to forward events to Cribl Edge.
   - The `IssuerCA` thumbprint is the SHA1 thumbprint of the issuing certificate, which could be an intermediate certificate. This must match the thumbprint of the first certificate in the certificate chain configured in the WEF Source's CA certificate path or the specified **CA fingerprint override**.
5. When complete, save the policy and apply it to affected endpoints. Link the GPO to the appropriate OU if it is not already linked.
6. To enable the Network Service to read the security logs, you'll need to configure access for the Event Log Service.



When configuring a Group Policy Object (GPO) to add NETWORK SERVICE permissions, carefully review existing permissions to prevent unintended removal. Preserve all necessary permissions to avoid application issues.

1. Open the Local Group Policy Editor tool (`gpedit.msc`).
2. Navigate to **Computer Configuration** > **Policies** > **Administrative Templates** > **Windows Components** > **Event Log Service**. Double-click **Security**, then in the **Settings** pane, select **Configure log access**.



Configuring log access

3. In the resulting modal, under **Options** > **Log Access**, enter the following Log Access configuration:

```
O:BAG:SYD:(A;;0xf0007;;;SY)(A;;0x7;;;BA)(A;;0x1;;;BO)(A;;0x1;;;SO)(A;;0x1;;;S-1-
```

Copy

4. Then, reboot your Windows machine to apply this setting. (Source: Microsoft's Security event log forwarding fails... topic.)

7. Check that events are flowing into Cribl Edge now according to the configured subscriptions. On client machines, run `gpupdate /force` to apply the new policy immediately. If events are not flowing, verify all of the following:
   - The clients can reach the Cribl Edge instance through the network to port `5986` (or other configured port) via TLS/HTTP. If clients are connecting to Cribl Stream via a proxy, you may need to toggle **Show originating IP** on (in the Advanced section of the Source configuration). Ensure that the correct outbound firewall port is opened on the client.
   - The certificate chain is correct
   - The endpoints have a valid CRL encompassing the CA cert
   - The CA cert is a trusted root on the clients

- The server and client certs are issued by the same CA. The `CAPI2` Windows event log might reveal any errors here.
- Check Cribl Edge for any errors, as well as the `EventForwarding-Plugin` and `Windows Remote Management` event logs on the clients.

Now you can provide client certificate authentication to a Windows Event Forwarder Source.

12.2.6.1.2. Kerberos Authentication For Windows Event Forwarder

This document covers the Kerberos authentication setup steps you need to take before configuring a Windows Event Forwarder Source.



Kerberos authentication is only supported on x86_64 Linux systems.

Initial Assumptions

To ensure a smooth configuration process, review the following prerequisites:

- The clients that will be sending events are in a Windows domain. In this example, the domain is `contoso.com`, the NETBIOS name is `CONTOSO`, and the domain controller is `DC01.contoso.com`.
- Kerberos KDC is already configured and working on the domain, and WEC with Kerberos authentication already works on the domain.
- You have an x86_64 Edge Node to which you have shell access. This guide assumes a recent Ubuntu distribution.
- The fully qualified domain name (FQDN) of this example Edge Node is `worker1.contoso.com`.
- You are using a Kerberos credential cache of the `DIR` or `FILE` type.

1. Initial Setup for the Edge Node

To set up the Edge Node:

1. Ensure that the Edge Node can resolve the domain controller's FQDN. You can test it with: `ping DC01.contoso.com`.
2. Ensure that the Edge Node's `hostname` is set to the FQDN. If it's not, you can set it with `hostnamectl set-hostname worker1.contoso.com`.
3. Ensure the Edge Node has its clock synchronized with the domain controller.
4. Install the `krb5-user` package and configure Kerberos to correctly identify your domain. Here's an example configuration (`/etc/krb5.conf`).

```
[libdefaults]
default_realm = CONTOSO.COM

[realms]
  CONTOSO.COM = {
    kdc = DC01.contoso.com
    admin_server = DC01.contoso.com
  }

[domain_realm]
  .contoso.com = CONTOSO.COM
  contoso.com = CONTOSO.COM
```

Copy

2. Setup for the Domain Controller

Configure the domain controller to support Kerberos authentication for your Edge Node. This involves setting up DNS records and creating a domain user with the necessary permissions and encryption settings. Proper configuration ensures that the Edge Node can securely authenticate and communicate within your network.

To set up the domain controller:

1. Set up forward and reverse DNS for the Edge Node's FQDN (that is, both A and PTR records).
2. Both `nslookup worker1.contoso.com` and `nslookup <worker's IP>` should work.
3. Create a domain user for the Edge Node. This guide assumes the user is called `svc-stream-worker1`.
4. Set the user's password to never expire.
5. Under the user's **Properties** > **Account** tab, set both `This account supports Kerberos AES 128 bit encryption` and `This account supports Kerberos AES 256 bit encryption`.
6. Create a keytab file for the Edge Node. This file will contain the necessary credentials for the Edge Node to authenticate with the domain controller. See the Keytab Export section below for instructions on how to generate the keytab file.

Keytab Export

This section guides you through exporting a keytab file, which is essential for Kerberos authentication. The keytab file contains the necessary credentials for the Edge Node to authenticate with the domain controller. You will generate this file using the `ktpass` command and then transfer it to the Edge Node. This step is crucial for enabling secure, authenticated communication between your Edge Node and the domain controller.

To export the keytab:

1. Export a keytab file for this user, which will also create the appropriate service principal name (SPN). The keytab will be exported to the root of the user profile running the command.
2. Open an elevated PowerShell prompt on the domain controller.
3. Run the following ktpass command to generate the keytab file: `ktpass /princ http/worker1.contoso.com@CONTOSO.COM /pass <password> /mapuser CONTOSO\svc-stream-worker1 /crypto AES256-SHA1 /ptype KRB5_NT_PRINCIPAL /out worker1.contoso.com.keytab`
    - `ktpass`: This is the command-line utility used to create keytab files for Kerberos authentication.
    - `/princ http/worker1.contoso.com@CONTOSO.COM`: Specifies the principal name. In this case, it is `http/worker1.contoso.com@CONTOSO.COM`. The principal name is a unique identifier for the service.
    - `/pass <password>`: Specifies the password for the user account. Replace `<password>` with the actual password for the `svc-stream-worker1` user.
    - `/mapuser CONTOSO\svc-stream-worker1`: Maps the principal name to the specified domain user account. Here, `CONTOSO\svc-stream-worker1` is the domain user created for the Edge Node.
    - `/crypto AES256-SHA1`: Specifies the encryption type to be used. `AES256-SHA1` is a strong encryption algorithm.
    - `/ptype KRB5_NT_PRINCIPAL`: Specifies the principal type. `KRB5_NT_PRINCIPAL` indicates that the principal is a Kerberos principal.
    - `/out worker1.contoso.com.keytab`: Specifies the output file name for the keytab. The keytab file will be saved as `worker1.contoso.com.keytab`.



Windows SPN compatibility

The SPN prefix you use in the `ktpass /princ` parameter must match what your Windows clients use when connecting. The example here uses `http/worker1.contoso.com@CONTOSO.COM`, which works in most environments, but some newer Windows versions (such as Windows Server 2025) require `host/worker1.contoso.com@CONTOSO.COM` instead.

To maximize compatibility across Windows versions, register both `http/worker1.contoso.com` and `host/worker1.contoso.com` SPNs on the account running the Edge Node. If Kerberos authentication fails with the `http/` example, regenerate the keytab using `/princ host/worker1.contoso.com@CONTOSO.COM` so it matches the SPN your clients require.

4. Note the SPN created here (in this example, `http/worker1.contoso.com@CONTOSO.COM`) as you will need it later. This SPN is case-sensitive.
5. Copy the keytab file to the Cribl Edge Edge Node. Carefully note the file's location on the Edge Node, because you will need it later. The keytab command and output will resemble this example:

```
PS C:\Users\Administrator> ktpass /princ http/wefkerb.weftest.local@WEFTEST.LOCAL /pa
Targeting domain controller: SUP-DC01.weftest.local
Using legacy password setting method

Successfully mapped http/wefkerb.weftest.local@WEFTEST.LOCAL to cribl.
Key created.

Output keytab to wefkerb.weftest.local.keytab:
Keytab version: 0x502
keysize 110 http/wefkerb.weftest.local@WEFTEST.LOCAL ptype 1 (KRB5_NT_PRINCIPAL) vno
```

Copy



If you rotate passwords on this user service account, you'll need to export a new keytab file and merge or replace the keytab on each Edge Node. Microsoft describes a general method for this task. The Microsoft method is for a different service, but it explains the general principles. You could also use a tool like `msktutil`.

3. Setup for the Windows Event Forwarding Target

To set up the Windows event forwarding target, you'll need to create group policy objects (GPOs) for the subscription target, the WinRM service, and the NETWORK SERVICE.

Create a GPO for the Subscription Target

Create a Group Policy Object (GPO) to configure the subscription target for Windows Event Forwarding (WEF). This GPO will define the target server and the refresh interval for event subscriptions.

1. Access the UI **Computer Configuration** > **Administrative Templates** > **Windows Components** > **Event Forwarding** > **Configure target Subscription Manager**.
2. Set to `Enabled`.
3. Go to **Subscription Managers** > **Show**.
4. Add `Server=http://worker1.contoso.com:5985/wsman/SubscriptionManager/WEC,Refresh=60` to

the list.

Create a GPO for the WinRM Service

Create a Group Policy Object (GPO) to ensure that the Windows Remote Management (WinRM) service is automatically started on client machines. This is a crucial step for enabling remote management and communication between the clients and the Edge Node. By setting the WinRM service to start automatically, you ensure that the necessary remote management capabilities are always available.

1. Access the UI **Computer Configuration** > **Policies** > **Windows Settings** > **Security Settings** > **System Services** > **Windows Remote Management (WS-Management)**.
2. Set **Startup type** to `Automatic`.

Create a GPO for the NETWORK SERVICE

Create a Group Policy Object (GPO) to grant the NETWORK SERVICE account the necessary permissions to read event logs. This is particularly important if you plan to subscribe to the Security event log. Properly configuring these permissions ensures that the `NETWORK SERVICE` account can access and forward the required event logs to the Edge Node.

1. Access the UI **Computer Configuration** > **Policies** > **Windows Settings** > **Security Settings** > **Restricted Groups**.
2. In the right pane, right-click and select **Add Group**.
3. Enter `Event Log Readers` as the group name.
4. Under `Members of this group`, click **Add**.
5. Enter `NETWORK SERVICE` as the member name.



For the change to take effect, you'll have to restart computers after you apply this GPO.

12.2.6.2. Windows Event Logs Source

Cribl Edge supports collecting local Windows Event Logs.

For guidance on workflow, see our better practices doc: Collect Windows Event Logs and Metrics.

Type: **Pull** | TLS Support: **N/A** | Event Breaker Support: **No** | Uses Key-Value Store

Cribl Edge Workers support Windows Event Logs only when running on Windows, not on Linux.

You can collect standard event logs with the Windows Event Logs Source - Application, Security, and System - and any other event logs on the machine.

Configuring Cribl Edge to Collect Windows Event Logs

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Source** and select the Source you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). Select the Source you want. Next, select **Add Source**.
2. Configure the following under **General Settings**:
    - **Input ID**: Enter a unique name to identify this Source definition. If you clone this Source, Cribl Edge will add `-CLONE` to the original **Input ID**.
    - **Description**: Optionally, enter a description.
    - **Event logs**: Enter one or more event logs to collect. Click **Add log** to specify more event logs (for example, `Application` or `System`). To identify the name of the log to collect, you can use the Windows Server Event Viewer. For more information, see Locate Windows Event Logs on the Server.
3. Next, you can configure the following **Optional Settings**:
    - **Read mode**: Select **From last entry** (the default) to read only new events. Select **Entire Log** to read all of the historical events and new events.
    - **Event format**: Select JSON or XML as the event format. JSON is the default. You can change this setting at any time.

- Events include the `__winEvent` property in both XML and JSON format. This property is one of the default **Excluded fields** in the Cribl TCP Destination. For details, see `__winEvent` Property.
- **Render event message strings**: Toggle on to display rendered message strings in the XML and JSON event output.
- This toggle only appears when you toggle **Use Windows Tools** off.
  - Defaults to toggled on when JSON is the selected event format and off when XML is the event format.
  - Events will not contain a `Message` property when toggled off.
- **Tags**: Optionally, add tags that you can use to filter and group Sources in Cribl Edge's UI. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

4. Optionally, you can adjust the Processing and Advanced settings, or Connected Destinations outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

Processing Settings Fields

In this section, you can define new fields or modify existing ones using JavaScript expressions, similar to the Eval function.

- The **Field Name** can either be a new field (unique within the event) or an existing field name to modify its value.
- The **Value** is a JavaScript expression (enclosed in quotes or backticks) to compute the field's value (can be a constant). Select this field's advanced mode icon (far right) if you'd like to open a modal where you can work with sample data and iterate on results.

This flexibility means you can:

- Add new fields to enrich the event.
- Modify existing fields by overwriting their values.
- Compute logic or transformations using JavaScript expressions.

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline or Pack to process data from this input before the data is sent through the Routes.

Advanced Settings

Use Windows Tools: Toggled off by default. This legacy setting will be removed in a future release. We highly recommend keeping this option disabled and using the newer native capabilities which are faster and more reliable.

- If toggled off:
    - The Windows Event Logs Source interacts directly with the Windows Event Log API, resulting in faster event log processing.
    - **Render event message strings** is available and allows you to include the `Message` property in JSON and XML events.
    - The event collector will drain the entire Windows event log (upon startup for the first time and when configured to read fully), or drain fully from the last place it read (upon subsequent reads and when Windows writes new events to the log). It will read as much as possible rather than in batches.
- If toggled on:
    - The Windows Event Logs Source will collect events using external tools like Powershell cmdlets or `wevtutil.exe`.
    - **Render event message strings** is not an option.
    - JSON events will contain rendered message strings, XML will not.

Polling interval: Specify how often, in seconds, to check for new event logs. Defaults to `10` seconds, and must be at least `1` second. Polling will read each log, up to the **Batch Size**, in their specified order. This Source will not make a polling request if it's still reading events from the previous request.

Batch size: Set the maximum number of events to read per polling request. By default, each request reads up to `500` events. Set as high as you need to; however, if you configure the Source to pull from multiple event logs, be aware that setting **Batch Size** higher can keep one log waiting longer while the Source collects a batch from another log.



Batch size and Polling interval only apply when **Use Windows Tools** is toggled on.

Max event bytes: The maximum number of bytes that can be in an event before Cribl Edge sends it to the Pipelines. Defaults to `51200`.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Connected Destinations

Send to Routes: Enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

QuickConnect: Send this Source's data to one or more Destinations via independent, direct connections.



This Source defaults to QuickConnect.

`__winEvent` Property

The `__winEvent` property contains a JSON representation of the native XML data obtained from the Windows Event API. The Windows Event Logs Source attempts to convert the JSON representation to the JSON schema that Windows Tools uses - however, there are some cases where the automatic conversion process doesn't perfectly match the JSON schema. The `__winEvent` property provides the raw JSON representation, allowing you to extract any necessary data if the conversion process isn't perfect.

Configure Event Log Behavior When Full

In Windows Event Viewer, you can configure how Windows handles Event Logs when the event log reaches maximum size.

To configure Windows Event Logs in **Event Viewer**, right-click on an event log and select **Properties**.

There are three available modes:

- **Overwrite events as needed**: When the log is full, Microsoft overwrites oldest events as new events enter.
- **Archive the log when full, do not overwrite events**: When the log is full, Microsoft rotates the file and starts a new file.
- **Do not overwrite events (Clear logs manually)**: When the log is full, Microsoft drops events until you clear the log manually.

In Cribl Edge, we designed the Windows Event Logs Source primarily for use with Event Logs in **Overwrite** mode, where it can collect events continuously. The WEL source may drop events in Event Logs when rotation occurs or when the log clears.

Collect From the Forwarded Events Logs

You can collect XML and JSON events from the Windows Forwarded Events log, as long as **Use Windows Tools** is toggled off in Advanced Settings.

Collect ETL-Formatted Logs

Microsoft Debug, Analytic, and Tracing logs use the ETL format. These logs are intended for short-term recording of detailed logging for troubleshooting and are not suitable for use with continuous collection. They cannot be read when set to Overwrite mode and they don't support seeking to skip past previously collected events.

The WEL Source in Edge supports reading events from ETL logs, but using them for continuous event collection may result in dropped events and higher resource usage.

Key-Value Store

The Windows Event Logs Source uses key-value store for cursors for each Windows Event Log. Log names are used as keys.

Resources

For processing Pipelines that you can import and adapt to your needs, examine these resources on the Cribl Packs Dispensary:

- Cribl Edge for Windows - Transform Windows logs collected by Cribl Edge to forward to various downstream services.
- Microsoft Windows Events - Shape events into JSON or Key=Value format, and dramatically reduce event sizes.
- Exabeam Pack for Windows - Forward specific Windows Event IDs to Exabeam.
- QRadar Pack for Windows Events - Optimize Windows events for delivery to QRadar.

Troubleshooting

If you are running this Source to collect events on a Windows Node without admin access, change the permission on the registry entry for the `EventLog` key: `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\EventLog` to `Read`.

If you encounter CPU or memory issue when running this Source, refer to Windows Troubleshooting.

Can't Collect ETL/Analytic/Debug Logs

When collecting ETL/analytic/debug log channels (such as `Microsoft-Windows-DNSServer/Analytical`), you can encounter a "The requested operation cannot be performed over an enabled direct channel" error.

The error is caused by the fact that this type of logs can't be read with the overwrite option enabled.

To resolve this:

1. Locate the log in Windows Event Viewer.
2. Right-click the log and select **Properties**.
3. Clear **Enable logging**.
   At this point Cribl Edge will be able to read the existing log entries, but will not receive any new entries added to the log.
4. To ensure Cribl Edge collects new entries as well, under **When maximum event log size is reached**, select **Do not overwrite events (Clear logs manually)**. (See Configure Event Log Behavior When Full for more details about those options.)
   Then, select **Enable logging** again.
5. Confirm your changes with **Apply**, then select **OK**.

You are now ready to ingest the log again with the Windows Event Logs Source.

12.2.6.3. Windows Metrics Source

Cribl Edge collects metrics from Windows hosts on which it runs, and can populate some standard metrics dashboards right out of the box. This Source produces events compatible with the Prometheus Windows Exporter to send out system, CPU, memory, network, and disk metrics. For metrics details, see Windows System Metrics Details.

Type: **System and Internal** | TLS Support: **N/A** | Event Breaker Support: **No**

Edge Nodes support Windows Metrics only when running on Windows. (To collect metrics from Linux hosts, use the System Metrics Source.)

Configure Cribl Edge/Windows to Collect System Metrics

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Collect** . Select **Add Source** and select the Source you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via Routes, select **More** > **Sources**. Select the Source you want. Next, select **Add Source**.
2. Configure the following under **General Settings**:
    - **Enabled**: Toggle on to enable the Source.
    - **Input ID**: Enter a unique name. The default Source is prefilled with the value `in_windows_metrics`, which can't be changed via the UI. If you clone this Source, Cribl Edge will add `-CLONE` to the original **Input ID**.
    - **Description**: Optionally, enter a description.
3. Next, you can configure the following **Optional Settings**:
    - **Polling interval**: How often, in seconds, to collect metrics. Defaults to `10` seconds.
    - **Tags**: Optionally, add tags that you can use for filtering and grouping in the Cribl Edge UI. Use a tab or hard return between (arbitrary) tag names. These tags aren't added to processed events.
4. Optionally, you can adjust the Host Metrics, Process Metrics, Processing and Advanced settings, or Connected Destinations outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

Host Metrics

Use the buttons to select a level of detail:

- **Basic** enables minimal metrics, averaged or aggregated. This is the minimum configuration needed to support the Cribl Edge landing page.
- **All** enables full, detailed metrics for individual CPUs, interfaces, etc.
- **Custom** displays submenus and buttons from which you can choose a level of detail (**Basic**, **All**, **Custom**, or **Disabled**) for each specific type of event.
- **Disabled** generates no metrics.

The meanings of **All** and **Disabled** are self-evident. **Basic** and **Custom** have different meanings depending on event type - see the following subsections.

System

Basic level captures load averages, uptime, and CPU count.

Custom level toggles **Detailed** metrics on or off. These are Windows-specific metrics including OS information, system uptime, CPU architecture, etc.

CPU

Basic level captures active, user, system, idle, and iowait percentages over all CPUs.

Custom level toggles the following on or off: **Per CPU metrics**, **Detailed metrics** (i.e., metrics for all CPU states), and **CPU time metrics** (i.e., raw, monotonic CPU time counters).

Memory

Basic level captures captures total, used, available, `swap_free`, and `swap_total`.

Custom level toggles **Detailed metrics** on or off. (These are metrics for all memory states.)

Network

Basic level captures bytes, packets, errors, and connections over all interfaces.

Custom level exposes the following:

- The **Interface filter**, which specifies which network interfaces to include or exclude. (An empty filter will include all metrics.)

- **Per-interface metrics**, which toggle on or off.
- **Detailed metrics**, which toggle on or off. If on, the **Protocol metrics** toggle appears, allowing you to choose whether to generate metrics for ICMP, ICMPMsg, IP, TCP, UDP, and UDPLite.

Disk

Basic level captures disk usage (%), bytes read and written, and read and write operations, over all mounted disks.

Custom level exposes the following:

- The **Volume filter**, specifying which Windows volumes to include or exclude. Supports wildcards and ! (not) operators. An empty filter will include all volumes.
- **Per-volume metrics**, which toggle on or off.
- **Detailed metrics**, which toggle on or off.

Process Metrics

With Process Metrics enabled, Cribl Edge captures process-specific metrics from Windows servers and reports them as events. This allows you to monitor specific processes on Cribl.Cloud instances. You can generate events for any process object.

To collect a process metrics event, create a Process Set and add a filter expression. Processes that match the filter are returned as individual events. See Collecting Metrics. Process Sets are separate from aggregate and host-wide metrics.

Process-specific metrics are **not** affected by the **Host Metrics** detail setting.

You can control the collection method for Process Metrics using a configuration option in the **Fleet Settings** > **Limits** > **Other** section. This setting allows you to choose between using PowerShell cmdlets or the more efficient native module. Be aware that when using PowerShell cmdlets, environment variables displayed in the **Explore** > **Processes** tab for Windows processes might be inaccurate. For details, see Other Limits.

Adding a Process Set

To add a Process Set:

1. Open the Windows Metrics Source and access the **Configure** tab.
2. Select the **Process Metrics** menu, then **Add Process Set**.
3. Configure the details:
    - **Set Name**: The name for this process set.
    - **Filter Expression**: The JavaScript expression that will filter the processes.

- **Include Child Processes**: When toggled on, the processes that match the filter include metrics for child processes.

Filtering Processes

You can filter processes using the field names and values from each process object.



Tip

To see all processes currently running on an Edge Node, select **Explore** in the submenu and open the **Processes Tab**.

Example Filters

Here's an example filter and the results it could return.

This filter will retrieve processes with the name `explorer`:

```
processName === 'explorer'
```

Note that the strict equality (`===`) operator in the above expression forces case sensitivity in the filter.



The **Filter expression** field expects a specific syntax in order to apply the filter to the processes. These expressions evaluate to either `true` or `false`. See Build Custom Logic to Route and Process Your Data: Filter Expressions for more information.

Collecting Metrics

Once you have at least one Process Set created and saved, Cribl Edge will begin collecting process metrics at the interval defined in the **Polling interval** field (**General Settings** menu).

To view the status of the Collector and total events collected, click the **Status** tab in the Windows Metrics Source.

To view a live capture of individual metrics gathered from processes that match the Process Set, click the **Live Data** tab. The Process Set that owns each metric is represented by the `__process_set` internal field.

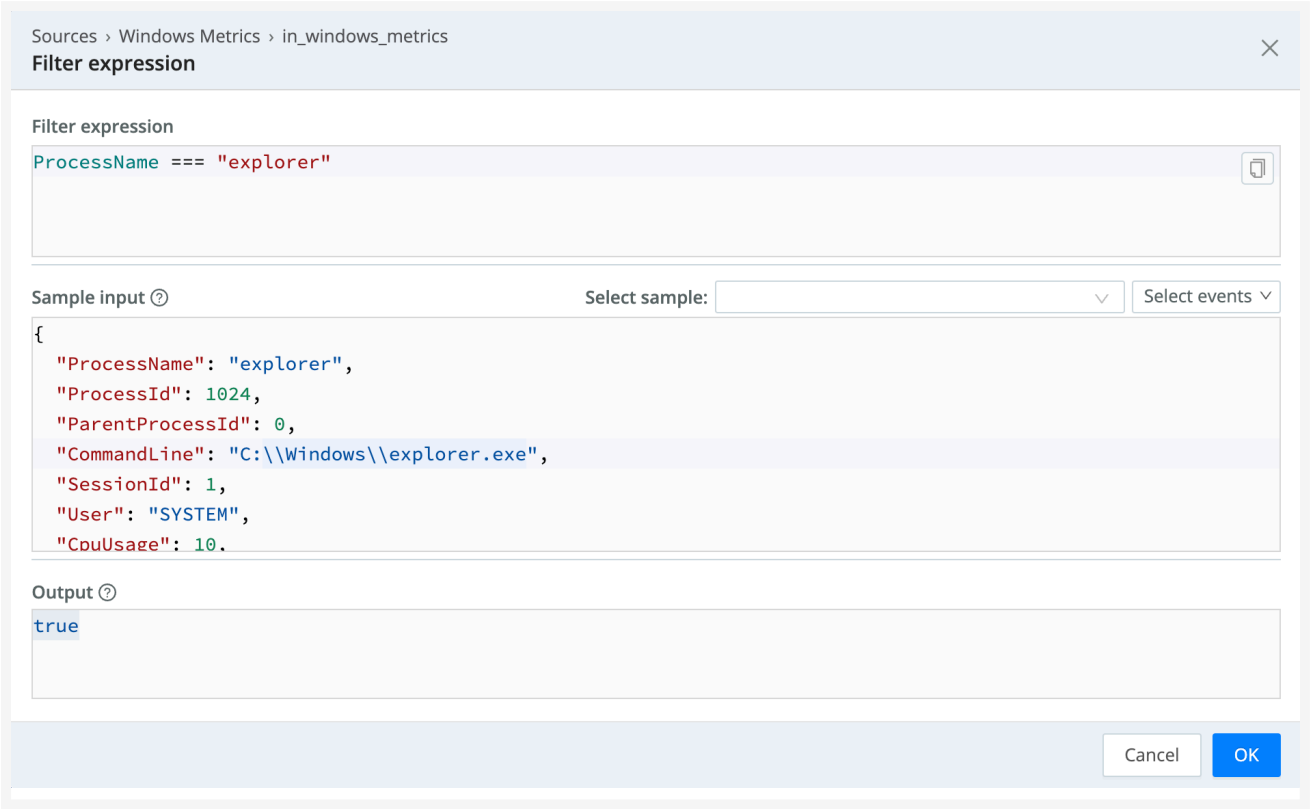
Supported Processes

To view the complete table of supported Windows process metrics, their descriptions, types, and dimensions, see Process-Specific Metrics.

Using Advanced Mode

To test your filter expression against a sample input, click the **Advanced mode** button within the **Filter expression** field. The **Filter expression** modal will open.

Here, you can paste in your filter expression, select a sample JSON input from the drop-down (or enter your own directly in the **Sample input** field), and select a specific event to test against.



The **Advanced mode** window, for process-specific metrics

The **Output** will show you if the filter expression returns any matching processes, based on the filter expressions and JSON input.

Processing Settings Fields

In this section, you can define new fields or modify existing ones using JavaScript expressions, similar to the Eval function.

- The **Field Name** can either be a new field (unique within the event) or an existing field name to modify its value.
- The **Value** is a JavaScript expression (enclosed in quotes or backticks) to compute the field's value (can be a constant). Select this field's advanced mode icon (far right) if you'd like to open a modal where you can work with sample data and iterate on results.

This flexibility means you can:

- Add new fields to enrich the event.
- Modify existing fields by overwriting their values.
- Compute logic or transformations using JavaScript expressions.

Pre-Processing

Select a **Pipeline** (or Pack) from the drop-down to process this Source's data. Required to configure this Source via **Data Routes**; optional to configure via **Collect/QuickConnect**.

Disk Spooling

Enable disk persistence: Whether to save metrics to disk. When toggled on, exposes this section's remaining fields.

Bucket time span: The amount of time that data is held in each bucket before it's written to disk. The default is 10 minutes (`10m`).

Max data size: Maximum disk space the persistent metrics can consume. Once reached, Cribl Edge will delete older data. Example values: `420 MB`, `4 GB`. Default value: `100 MB`.

Max data age: How long to retain data. Once reached, Cribl Edge will delete older data. Example values: `2h`, `4d`. Default value: `24h` (24 hours).

Compression: Optionally compress the data before sending. Defaults to `gzip` compression. Select `none` to send uncompressed data.

Path location: Path to write metrics to. Default value is `$CRIBL_HOME/state/windows_metrics`.

Advanced Settings

Use Windows Tools: Toggled off by default. This legacy setting will be removed in a future release. We highly recommend keeping this option disabled and using the newer native capabilities which are faster and more reliable.

- If toggled off, the Windows Metrics Source interacts directly with the Windows Management Instrumentation (WMI) API, resulting in faster metrics collection.
- If toggled on, the Windows Metrics Source will collect events using Powershell cmdlets.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

12.2.7. AppScope Source

AppScope is deprecated and will be removed in a future release. Don't use AppScope in production and ensure it is not part of your data flow configuration. For details, please read the announcement in Cribl Community Slack.

12.2.8. Cloudflare Source

The Cloudflare Source receives logs and events directly from Cloudflare Logpush using the HTTP Event Collector (HEC) protocol. This Source is designed for seamless integration with the Cloudflare push API, supporting both structured and raw event formats.

Type: **Push** | TLS Support: **YES** | Event Breaker Support: **YES**

This Source supports gzip-compressed inbound data when the `Content-Encoding: gzip` connection header is set.

Configure a Cloudflare Source

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Source** and select the Source you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). Select the Source you want. Next, select **Add Source**.
2. Configure the following under **General Settings**:
    - **Input ID**: Enter a unique name to identify this Source definition. If you clone this Source, Cribl Edge will add `-CLONE` to the original **Input ID**.
    - **Description**: Optionally, enter a description.
    - **Address**: Enter the hostname/IP on which to listen for HTTP(S) data. Such as: `localhost` or `0.0.0.0`. Supports IPv4 and IPv6 addresses.
    - **Port**: Enter the port number.
    - **HEC endpoint**: Absolute path on which to listen for the Cloudflare HTTP Event Collector API requests. Defaults to `/services/collector`. This single endpoint supports JSON events via `/event` and raw events via `/raw`.
3. Next, you can configure the following **Optional Settings**:
    - **Allowed Indexes**: List the values allowed in the HEC event index field. Allows wildcards. Leave blank to skip validation.

- **Tags**: Add tags that you can use to filter and group Sources in Cribl Edge's UI. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

4. Optionally, you can adjust the TLS, Persistent Queue Settings, Processing, Auth tokens, and Advanced settings, or Connected Destinations outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

TLS Settings (Server Side)

Enabled: Defaults to toggled off. When toggled on:

Certificate: Name of the predefined certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

Passphrase: Passphrase to use to decrypt private key.

Certificate path: Server path containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

CA certificate path: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

Authenticate client (mutual auth): Require clients to present their certificates. Used to perform mutual authentication using SSL certs. Default is toggled off. When toggled on:

- **Validate client certificates**: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (such as the system's CA). Default is toggled off.
- **Common name**: Regex that a peer certificate's subject attribute must match in order to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. (For example, to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.) If the subject attribute contains Subject Alternative Name (SAN) entries, the Source will check the regex against all of those but ignore the Common Name (CN) entry (if any). If the certificate has no SAN extension, the Source will check the regex against the single name in the CN.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In the **Persistent Queue Settings** tab, you can optionally specify persistent queue storage, using the following controls. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Source Persistent Queues (sPQ)

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the **Enable persistent queue** toggle. If enabled, PQ is automatically configured in `Always On` mode, with a maximum queue size of 1 GB disk space allocated per PQ-enabled Source, per Worker Process.

The 1 GB limit is on uncompressed inbound data, and the queue does not perform any compression. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable persistent queue: Default is toggled off. When toggled on:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to the Cribl Stream data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from the Cribl Stream data processing engine.

`Smart` mode only engages when necessary, such as when a downstream Destination becomes blocked *and* the **Buffer size limit** reaches its limit. When persistent queue is set to `Smart` mode, Cribl attempts to flush the queue when every new event arrives. The only time events stay in the buffer is when a downstream Destination becomes blocked.

Buffer size limit: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. This buffer is for all connections, not just per Worker Process. For that reason, this can dramatically expand memory usage. Connections share this limit, which may result in slightly lower throughput for higher numbers of connections. For higher numbers of connections, consider increasing the limit.

Commit frequency: The number of events to send downstream before committing that Stream has read them. Defaults to `42`.

File size limit: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, and so forth. If not specified, Cribl Stream applies the default `1 MB`.

Queue size limit: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data. Required, and defaults to `5 GB`. Accepts positive numbers with units of `KB`, `MB`, `GB`, and so forth. Can be set as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** in Group or Fleet settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file closes. Defaults to `None`; `Gzip` is also available.



In Cribl Stream 4.1 and newer, the Source persistent queue default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Optimize Source Persistent Queues (sPQ).

You can optimize Workers' startup connections and CPU load at **Group/Fleet settings** > Worker Processes.

Processing Settings Event Breakers

Cloudflare Source Event Breakers only function with the `/raw` endpoint. The `/event` endpoint operates on the assumption that the data they receive is already properly formatted into individual events and therefore do not utilize Event Breakers.

- `/raw`: Event Breakers split incoming raw, unstructured data into distinct events before processing them through Pipelines.
- `/event`: Event Breakers are not applied since data is already segmented in JSON format.

Event Breaker rulesets: A list of event breaking rulesets that will be applied to the input data stream before the data is sent through the Routes. Defaults to `System Default Rule`.

Event Breaker buffer timeout: How long (in milliseconds) the Event Breaker will wait for new data to be sent to a specific channel, before flushing out the data stream, as-is, to the Routes. Minimum `10` ms, default `10000` (10 sec), maximum `43200000` (12 hours).

Fields

In this section, you can add Fields to each event using Eval-like functionality.

Name: Field name.

Value: JavaScript expression to determine the field's value (can be a constant).

Fields specified on the **Fields** tab will normally override fields of the same name in events. But you can specify that fields in events should override these fields' values.

In particular, where incoming events have no `index` field, this Source adds one with the literal value `default`. You can override this value by using **Add Field** to specify an `index` field, and then setting its **Value** to an expression of the following form: `index == 'default' ? 'myIndex' : index`

Fields here are evaluated and applied **after** any fields specified in the Auth Tokens section.

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline or Pack to process data from this input before the data is sent through the Routes.

Auth Tokens

Provides the authentication token required to access this Source's data. Ensure the token is valid and has the necessary permissions for data retrieval.

Not providing a token allows **unauthorized access**.

Authentication method: Choose between manually entering a **Token** or selecting a stored **Secret**. If you opt for manual entry, type the token directly or **Generate** a new one. For added security, use the secrets store to select a stored **Token (text secret)** or **Create** a new one.

Enable token: Controls the status of the specified authentication token. When toggled on, the specified auth token is active and can be used for client access. If toggled off, the token is disabled, causing it to be skipped and ignored during initialization. This means that any client attempting to use a disabled token will not be granted access.

Disabling a token can be helpful when you need to temporarily stop receiving data from a particular service or application. Disabling a token prevents the service or application from sending data to your endpoint without having to completely delete the token.

See also Periodic Logging for information on how auth tokens affect product logging.

Description: Optional (but recommended), the description for this token. Although this field is optional, the token description becomes the display name for the token. You can select this display name to expand or collapse the token details.

Allowed indexes: Optionally, specify which indexes are permissible for events ingested using this token. Supports multiple index values and wildcards.

Fields: Fields to add to events referencing this token. Each field is a **Name/Value** pair.



Fields specified on the **Auth Tokens** tab will normally override fields of the same name in events. But you can specify that fields in events should override these fields' values.

In particular, where incoming events have no `index` field, this Source adds one with the literal value `default`. You can override this value by using **Add Field** to specify an `index` field, and then setting its **Value** to an expression of the following form: `index == 'default' ? 'myIndex' : index`

Fields here are evaluated and applied **before** any fields specified in the Fields section.

Advanced Settings

Emit per-token request metrics: Toggle on to emit per-token `<prefix>.http.perToken` and summary `<prefix>.http.summary` request metrics. This enables Periodic Logging.

Show originating IP: Toggle on when clients are connecting through a proxy that supports the `X-Forwarded-For` header to keep the client's original IP address on the event instead of the proxy's IP address. This setting affects how the Source handles the `__srcIpPort` field.

Capture request headers: Toggle on to add request headers to events, in the `__headers` field.

CORS allowed origins: If you need to enable cross-origin resource sharing, use this field to specify the HTTP origins to which Cribl Edge should send `Access-Control-Allow-*` headers. You can enter multiple domains and use wildcards. This and its companion **CORS allowed headers** option should seldom be needed - see Working with CORS Headers for details.

CORS allowed headers: List HTTP headers that Cribl Edge will send in a CORS preflight response to your configured **CORS allowed origins** as `Access-Control-Allow-Headers`. Enter `*` to allow all headers.

Active request limit (on-prem only): Maximum number of active requests allowed for this Source, per Worker Process. Defaults to `256`. Enter `0` for unlimited.



Raising this limit can increase throughput by allowing more concurrent data requests, but increases resource usage and load on both your Cribl Edge infrastructure and on downstream Destinations. Before raising this limit, ensure:

- Your Cribl Edge deployment has sufficient capacity to support higher request concurrency, including CPU, memory, or number of Worker Processes.
- Downstream Destinations are correctly sized and tuned to accept the higher data ingest rate, preventing backpressure. See Manage Backpressure for more information.

> Improper sizing on either side can result in dropped events, delayed processing, or overall system instability.

Activity log sample rate: Determines how often request activity is logged at the `info` level. The default `100` value logs every 100th value; a `1` value would log every request; a `10` value would log every 10th request; and so forth.

Requests-per-socket limit: The maximum number of requests Cribl Edge should allow on one socket before instructing the client to close the connection. Defaults to `0` (unlimited). See Balancing Connection Reuse Against Request Distribution below.

Socket timeout (seconds): How long Cribl Edge should wait before assuming that an inactive socket has timed out. The default `0` value means wait forever.

Request timeout (seconds): How long to wait for an incoming request to complete before aborting it. The default `0` value means wait indefinitely.

Keep-alive timeout (seconds): After the last response is sent, Cribl Edge will wait this long for additional data before closing the socket connection. Defaults to `5` seconds; minimum is `1` second; maximum is `600` seconds (10 minutes).

> The longer the **Keep-alive timeout**, the more Cribl Edge will reuse connections. The shorter the timeout, the closer Cribl Edge gets to creating a new connection for every request. When request frequency is high, you can use longer timeouts to reduce the number of connections created, which mitigates the associated cost.

IP allowlist regex: Grants access to requests originating from specific IP addresses that match a defined pattern. Unmatched requests are rejected with a 403 (Forbidden) status code. Defaults to `.*` (allow all).

IP denylist regex: Blocks requests originating from specific IP addresses that match a defined pattern, even if they would be allowed by default. Rejected requests receive a 403 (Forbidden) status code. Defaults to `^$` (allow all).

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Balancing Connection Reuse Against Request Distribution

Requests-per-socket limit allows you to limit the number of HTTP requests an upstream client can send on one network connection. Once the limit is reached, Cribl Edge uses HTTP headers to inform the client that it must establish a new connection to send any more requests. (Specifically, Cribl Edge sets the HTTP `Connection` header to `close`.) After that, if the client disregards what Cribl Edge has asked it to do and tries

to send another HTTP request over the existing connection, Cribl Edge will respond with an HTTP status code of `503 Service Unavailable`.

Use this setting to strike a balance between connection reuse by the client, and distribution of requests among one or more Edge Node processes by Cribl Edge:

- When a client sends a sequence of requests on the same connection, that is called connection reuse. Because connection reuse benefits client performance by avoiding the overhead of creating new connections, clients have an incentive to **maximize** connection reuse.
- Meanwhile, a single process on that Edge Node will handle all the requests of a single network connection, for the lifetime of the connection. When receiving a large overall set of data, Cribl Edge performs better when the workload is distributed across multiple Edge Node processes. In that situation, it makes sense to **limit** connection reuse.

There is no one-size-fits-all solution, because of variation in the size of the payload a client sends with a request and in the number of requests a client wants to send in one sequence. Start by estimating how long connections will stay open. To do this, multiply the typical time that requests take to process (based on payload size) by the number of requests the client typically wants to send.

If the result is 60 seconds or longer, set **Requests-per-socket limit** to force the client to create a new connection sooner. This way, more data can be spread over more Edge Node processes within a given unit of time.

For example: Suppose a client tries to send thousands of requests over a very few connections that stay open for hours on end. By setting a relatively low **Requests-per-socket limit**, you can ensure that the same work is done over more, shorter-lived connections distributed between more Edge Node processes, yielding better performance from Cribl Edge.

A final point to consider is that one Cribl Edge Source can receive requests from more than one client, making it more complicated to determine an optimal value for **Requests-per-socket limit**.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

Internal Fields

Cribl Edge uses a set of internal fields to assist in handling of data. These "meta" fields are **not** part of an event, but they are accessible, and Functions can use them to make processing decisions.

Fields for this Source:

- `__headers` - Added only when **Advanced Settings** > **Capture request headers** is toggled on.
- `__hecToken`
- `__inputId`
- `__ctrlFields`
- `__srcIpPort` - See Overriding `__srcIpPort` with Client IP/Port for details.
- `__TZ`

Overriding `__srcIpPort` with Client IP/Port

The `__srcIpPort` field's value contains the IP address and (optionally) port of the client sending data to this Source.

When any proxies (including load balancers) lie between the client and the Source, the last proxy adds an `X-Forwarded-For` header whose value is the IP/port of the original client. With multiple proxies, this header's value will be an array, whose first item is the original client IP/port.

If `X-Forwarded-For` is present, and **Advanced Settings** > **Show originating IP** is toggled off, the original client IP/port in this header will override the value of `__srcIpPort`.

If **Show originating IP** is toggled on, the `X-Forwarded-For` header's contents will **not** override the `__srcIpPort` value. (Here, the upstream proxy can convey the client IP/port without using this header.)

Working with CORS Headers

The **CORS allowed origins** and **CORS allowed headers** settings are needed **only** when you want JavaScript code running in a web browser to send events to the Source. In this (uncommon) scenario, the code in question sends a preflight request that includes CORS (Cross-Origin Resource Sharing) headers, and the Source includes CORS headers in its response.

If the settings (and thus the CORS headers in the response) are correct, the browser will allow the code in question to complete requests to the Source - that is, allow the code to read the Source's responses. (These settings are optional because for some web applications it's sufficient to send requests **without** reading responses.)

The CORS header dance begins when the web browser sends a request with a header stating its origin. What happens next depends on how you've configured the CORS header settings, as explained in the following tables. (For these examples, we'll assume an origin of `https://mycoolwebapp:3001`.)

CORS allowed origins value	What Source does	Does Browser then allow code to read response?
`*`	Send `Access-Control-Allow-Origin: *` header.	Yes
Multiple wildcard values	If origin value matches any wildcard, send `Access-Control-Allow-Origin: https://mycoolwebapp:3001` header.	Yes
Nothing (not configured)	Send neither an `Access-Control-Allow-Origin` nor an `Access-Control-Allow-Headers` header.	No

The web browser's request might also include an `Access-Control-Request-Headers` header that lists one or more headers that it wants to use in a subsequent `POST` request (that is, the "actual" as opposed to preflight request). The Source can then respond with an `Access-Control-Allow-Headers` header **if and only if** it's also sending the `Access-Control-Allow-Origin` header.

CORS allowed headers value	What Source does	Does Browser then allow code to read response?
`*`	Send `Access-Control-Allow-Headers` header with the same list the client sent.	Yes
Multiple wildcard values	If any headers from the client's list match wildcards, send `Access-Control-Allow-Headers` header listing the matching headers.	Yes
Nothing (not configured)	Do not send an `Access-Control-Allow-Headers` header.	Probably not

Periodic Logging

Cribl Edge emits metrics about incoming requests when **Advanced Settings > Emit per-token request metrics** is enabled. These metrics can be routed to a Destination by enabling the Cribl Metrics Source. There are two categories of metrics: per-token metrics provide details for each configured token, while summary metrics aggregate data across all tokens. If no tokens are configured, only summary metrics will be emitted.

Per-token metric names will be prefixed by `<basePrefix>.http.perToken.`, where `<basePrefix>` is the configured base prefix for the Cribl Metrics Source. The available metric names are listed below, excluding the prefix. As these are counter-type metrics, they will not be emitted if the value is zero.

- `bytesReceived`
- `numRequests`

- `numEvents`
- `numErrors` *Note: this is cumulative of specific error types below as well as a catch-all for other types*
- `numParserErrors`
- `numToDisabledToken`

The summary metric names will be prefixed by `<basePrefix>.http.summary.` where `<basePrefix>` is the configured base prefix for the Cribl Metrics Source. The available metric names are listed below, excluding the prefix. As these are counter-type metrics, they will not be emitted if the value is zero.

- `bytesReceived`
- `numRequests`
- `numEvents`
- `numErrors` *Note: this is cumulative of specific error types below as well as a catch-all for other types*
- `numParserErrors`
- `numToDisabledToken`
- `numAuthFailures`
- `numToWrongUrl`
- `numAckReqs`
- `numReqsAcked`

Troubleshooting

The Source's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they are ingested. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the ingestion process, including any errors or warnings that may have occurred.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify ingestion issues. Analyze the graphs showing events and bytes in/out over time.

Common Issues

Dropping request because token invalid"."authToken": "Bas...Nic="

12.2.9. Datadog Agent Source

Datadog Agent is open-source software that monitors the host on which it runs. Acting as a DogStatsD server, Datadog Agent also aggregates metrics from other processes or containers on the host.

Type: **Push** | TLS Support: **YES** | Event Breaker Support: **No**

Looking for the Datadog Agent Source in **Cribl Search**? See Ingest Datadog Agent Data into Cribl Search.

Cribl Edge can ingest the following from Datadog Agent, in bundled form:

- Logs.
- Metrics (gauge, rate, counter, and histogram).
- Service checks.
- Agent metadata and other events emitted from the `/intake/` endpoint.

APM traces: While the Datadog Agent typically sends APM data to Datadog by default, this Source cannot receive APM data routed through the Datadog Agent (via `DD_APM_DD_URL`). Instead, it can receive APM traces directly from applications that use the Datadog tracing libraries (`dd-trace`) when those applications are configured to send trace data to this Source using the Datadog trace API v0.3 or v0.4 endpoints. For details on instrumenting your applications, see the Datadog tracing library documentation.

On the system(s) that you want to monitor, you'll need to install Datadog Agent and configure it to send data to Cribl Edge. Cribl Edge can parse, filter, and enrich that data, and then send it to any supported Destination, including a Cribl Edge Datadog Destination. (By default, Datadog Agent sends data only to Datadog.)

For inbound log data, this Source supports gzip -compression when the `Content-Encoding: gzip` connection header is set. For other data types, it assumes that all inbound data is compressed using `deflate`.

Configure Datadog to use Datadog API v1

This Source communicates with the Datadog API. Although v2 is the latest version of this API, this Source uses v1, which Datadog still supports.

To configure your Datadog Agent to work with this Source, ensure that in the `datadog.yaml` file, `use_v2_api.series` is set to `false`. Otherwise, when Datadog Agent sends metrics, Cribl Edge will not receive them, and `API Key invalid, dropping transaction` errors will appear in the Datadog Agent logs.

> Potential Cribl Edge support for Datadog API v2 is being tracked as **CRIBL-18281** in Known Issues.

Configure Cribl Edge to Ingest Datadog Agent Output

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Source** and select the Source you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). Select the Source you want. Next, select **Add Source**.
2. In the **New Source** modal, configure the following under **General Settings**:
    - **Input ID**: Enter a unique name to identify this Source definition. If you clone this Source, Cribl Edge will add `-CLONE` to the original **Input ID**.
    - **Description**: Optionally, enter a description.
    - **Address**: Enter hostname/IP to listen for Datadog Agent data. For example, `localhost` or `0.0.0.0`.
    - **Port**: Enter the port number to listen on.
3. Next, you can configure the following **Optional Settings**:
    - **Tags**: Optionally, add tags that you can use to filter and group Sources in Cribl Edge's UI. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
4. Optionally, you can adjust the TLS, Persistent Queue Settings, Processing and Advanced settings, or Connected Destinations outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

TLS Settings (Server Side)

Enabled: Defaults to toggled off. When toggled on:

Certificate: Name of the predefined certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

Passphrase: Passphrase to use to decrypt private key.

Certificate path: Server path containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

CA certificate path: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

Authenticate client (mutual auth): Require clients to present their certificates. Used to perform mutual authentication using SSL certs. Default is toggled off. When toggled on:

- **Validate client certificates**: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (for example, the system's CA). Default is toggled on.
- **Common name**: Regex that a peer certificate's subject attribute must match in order to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. (For example, to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.) If the subject attribute contains Subject Alternative Name (SAN) entries, the Source will check the regex against all of those but ignore the Common Name (CN) entry (if any). If the certificate has no SAN extension, the Source will check the regex against the single name in the CN.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In the **Persistent Queue Settings** tab, you can optionally specify persistent queue storage, using the following controls. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Source Persistent Queues (sPQ)

> On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the **Enable persistent queue** toggle. If enabled, PQ is automatically configured in `Always On` mode, with a maximum queue size of 1 GB disk space allocated per PQ-enabled Source, per Worker Process.
>
> The 1 GB limit is on uncompressed inbound data, and the queue does not perform any compression. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable persistent queue: Default is toggled off. When toggled on:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to the Cribl Stream data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from the Cribl Stream data processing engine.



`Smart` mode only engages when necessary, such as when a downstream Destination becomes blocked *and* the **Buffer size limit** reaches its limit. When persistent queue is set to `Smart` mode, Cribl attempts to flush the queue when every new event arrives. The only time events stay in the buffer is when a downstream Destination becomes blocked.

Buffer size limit: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. This buffer is for all connections, not just per Worker Process. For that reason, this can dramatically expand memory usage. Connections share this limit, which may result in slightly lower throughput for higher numbers of connections. For higher numbers of connections, consider increasing the limit.

Commit frequency: The number of events to send downstream before committing that Stream has read them. Defaults to `42`.

File size limit: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, and so forth. If not specified, Cribl Stream applies the default `1 MB`.

Queue size limit: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data. Required, and defaults to `5 GB`. Accepts positive numbers with units of `KB`, `MB`, `GB`, and so forth. Can be set as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** in Group or Fleet settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file closes. Defaults to `None`; `Gzip` is also available.



In Cribl Stream 4.1 and newer, the Source persistent queue default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Optimize Source Persistent Queues (sPQ).

You can optimize Workers' startup connections and CPU load at **Group/Fleet settings** > Worker Processes.

Processing Settings Fields

In this section, you can define new fields or modify existing ones using JavaScript expressions, similar to the Eval function.

- The **Field Name** can either be a new field (unique within the event) or an existing field name to modify its value.
- The **Value** is a JavaScript expression (enclosed in quotes or backticks) to compute the field's value (can be a constant). Select this field's advanced mode icon (far right) if you'd like to open a modal where you can work with sample data and iterate on results.

This flexibility means you can:

- Add new fields to enrich the event.
- Modify existing fields by overwriting their values.
- Compute logic or transformations using JavaScript expressions.

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline or Pack to process data from this input before the data is sent through the Routes.

Advanced Settings

Show originating IP: Toggle on when clients are connecting through a proxy that supports the `X-Forwarded-For` header to keep the client's original IP address on the event instead of the proxy's IP address. This setting affects how the Source handles the `__srcIpPort` field.

Capture request headers: Toggle on to add request headers to events, in the `__headers` field.

Health check endpoint: Toggle on to enable a health check endpoint specific to this Source, `http(s)://<host>:<port>/cribl_health`. A `200` HTTP response code is returned when the Source is healthy. Otherwise, two errors you could receive are:

- `ECONNRESET` where the Source failed to initialize due to not having listeners on the port.
- `503` or `Server is busy, max active connections reached` indicate there are too many connections per Worker Process.

Active request limit (on-prem only): Maximum number of active requests per Worker Process. Use `0` for unlimited.



Raising this limit can increase throughput by allowing more concurrent data requests, but increases resource usage and load on both your Cribl Edge infrastructure and on downstream Destinations. Before raising this limit, ensure:

- Your Cribl Edge deployment has sufficient capacity to support higher request concurrency, including CPU, memory, or number of Worker Processes.
- Downstream Destinations are correctly sized and tuned to accept the higher data ingest rate, preventing backpressure. See Manage Backpressure for more information.

Improper sizing on either side can result in dropped events, delayed processing, or overall system instability.

Requests-per-socket limit: The maximum number of requests Cribl Edge should allow on one socket before instructing the client to close the connection. Defaults to `0` (unlimited). See Balancing Connection Reuse Against Request Distribution below.

Socket timeout (seconds): How long Cribl Edge should wait before assuming that an inactive socket has timed out. The default `0` value means wait forever.

Keep-alive timeout (seconds): After the last response is sent, Cribl Edge will wait this long for additional data before closing the socket connection. Defaults to `5` seconds; minimum is `1` second; maximum is `600` seconds (10 minutes).

Extract metrics: Toggle on to extract each incoming metric to multiple events, one per data point. This works well when sending metrics to a `statsd`-type output. If sending metrics to DatadogHQ or any destination that accepts arbitrary JSON, leave toggled off (default).

Forward API key validation requests: Toggle on to send key validation requests from Datadog Agent to the Datadog API. If toggled off (the default), Stream handles key validation requests by always responding that the key is valid.

Reject unauthorized certificates: Reject certificates that are not authorized by a trusted CA (for example, the system's CA). Default is toggled on. Available when **Forward API key validation requests** is toggled on.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Balancing Connection Reuse Against Request Distribution

Requests-per-socket limit allows you to limit the number of HTTP requests an upstream client can send on one network connection. Once the limit is reached, Cribl Edge uses HTTP headers to inform the client that it must establish a new connection to send any more requests. (Specifically, Cribl Edge sets the HTTP `Connection` header to `close`.) After that, if the client disregards what Cribl Edge has asked it to do and tries to send another HTTP request over the existing connection, Cribl Edge will respond with an HTTP status code of `503 Service Unavailable`.

Use this setting to strike a balance between connection reuse by the client, and distribution of requests among one or more Edge Node processes by Cribl Edge:

- When a client sends a sequence of requests on the same connection, that is called connection reuse. Because connection reuse benefits client performance by avoiding the overhead of creating new connections, clients have an incentive to **maximize** connection reuse.
- Meanwhile, a single process on that Edge Node will handle all the requests of a single network connection, for the lifetime of the connection. When receiving a large overall set of data, Cribl Edge performs better when the workload is distributed across multiple Edge Node processes. In that situation, it makes sense to **limit** connection reuse.

There is no one-size-fits-all solution, because of variation in the size of the payload a client sends with a request and in the number of requests a client wants to send in one sequence. Start by estimating how long connections will stay open. To do this, multiply the typical time that requests take to process (based on payload size) times the number of requests the client typically wants to send.

If the result is 60 seconds or longer, set **Requests-per-socket limit** to force the client to create a new connection sooner. This way, more data can be spread over more Edge Node processes within a given unit of time.

For example: Suppose a client tries to send thousands of requests over a very few connections that stay open for hours on end. By setting a relatively low **Requests-per-socket limit**, you can ensure that the same work is done over more, shorter-lived connections distributed between more Edge Node processes, yielding better performance from Cribl Edge.

A final point to consider is that one Cribl Edge Source can receive requests from more than one client, making it more complicated to determine an optimal value for **Requests-per-socket limit**.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

Internal Fields

Cribl Edge uses a set of internal fields to assist in handling of data. These "meta" fields are **not** part of an event, but they are accessible, and Functions can use them to make processing decisions.

Fields for this Source:

- `__agent_api_key`
- `__agent_event_type`
- `__final`

- `__headers` - Added only when **Advanced Settings** > **Capture request headers** is toggled on.
- `__inputId`
- `__srcIpPort` - See details below.
- `_time`

Overriding `__srcIpPort` with Client IP/Port

The `__srcIpPort` field's value contains the IP address and (optionally) port of the Datadog Agent sending data to this Source.

When any proxies (including load balancers) lie between the Datadog Agent and the Source, the last proxy adds an `X-Forwarded-For` header whose value is the IP/port of the original client. With multiple proxies, this header's value will be an array, whose first item is the original client IP/port.

If `X-Forwarded-For` is present, and **Advanced Settings** > **Show originating IP** is toggled off, the original client IP/port in this header will override the value of `__srcIpPort`.

If **Show originating IP** is toggled on, the `X-Forwarded-For` header's contents will **not** override the `__srcIpPort` value. (Here, the upstream proxy can convey the client IP/port without using this header.)

Sending Datadog Agent Data to Cribl Edge

Before you begin this section, you should have Datadog Agent running on one or more hosts.

To enable Datadog Agent to send data to Cribl Edge, you'll set the following environment variables:

- `DD_DD_URL`: The URL or IP address and port of your Datadog Agent Source in Cribl Edge.
- `DD_LOGS_CONFIG_LOGS_DD_URL`: The same value as above, assuming log collection is enabled on Datadog Agent.
- `DD_LOGS_CONFIG_USE_HTTP`: Set to `true`. Cribl Edge ingests Datadog Agent logs over HTTP only; ingesting logs over TCP is not supported.

Set these environment variables in one of two ways: (1) through a `docker run` command, or (2) by editing the `datadog.yaml` configuration file that your Datadog Agent uses.

Using the `docker run` Command

Here's an example `docker run` that sets the environment variables described above, along with others required for Datadog Agent but not relevant to Cribl Edge. Replace the example values with values appropriate for your environment.

```
docker run --rm --name dd-agent
  -e DD_API_KEY=6d1ephx1w55p978i6d1ephx1w55p978i \
  -e DD_SKIP_SSL_VALIDATION=true \
  -e DD_DD_URL="http://0.0.0.0:8080" \
  -e DD_LOGS_ENABLED=true \
  -e DD_LOGS_CONFIG_LOGS_DD_URL="0.0.0.0:8080" \
  -e DD_LOGS_CONFIG_USE_HTTP=true \
  -e DD_LOGS_CONFIG_LOGS_NO_SSL=true \
  -e DD_LOGS_CONFIG_CONTAINER_COLLECT_ALL=true \
  -e DD_DOGSTATSD_NON_LOCAL_TRAFFIC="true" \
  gcr.io/datadoghq/agent:7
```

Copy

Using a Config File

Instead of using environment variables, you can set the Cribl Edge-related values in your Datadog Agent's `datadog.yaml` config file.

See the documentation links in the example config file that Datadog maintains online; and, the docs that describe filesystem locations relevant to getting Datadog Agent to configure itself with the desired `datadog.yaml` file.

Managing What Data Goes Where

The different kinds of information that Datadog Agents emit - logs, metrics, service checks, agent metadata, and APM data - are not completely independent when you send them to Cribl Edge Datadog Agent Source. The reference to "bundling" near the top of this page introduced this situation in simplified form. The table below explains the relevant constraints for each type of information that Datadog Agents emit, along with the Datadog environment variables that control them.

Type(s) of Information	Dependencies and/or Limitations	Environment Variable
Logs	Independent of other types.	`DD_LOGS_CONFIG_DD_URL`
Metrics, Events, Service Checks, and Metadata	Always sent together.	`DD_DD_URL`

Type(s) of Information	Dependencies and/or Limitations	Environment Variable
APM Data (via Datadog Agent)	When configured with `DD_APM_DD_URL`, APM data can only be sent to Datadog (for example, `https://trace.agent.datadoghq.com`) and cannot be sent to Cribl Edge, even when you send other data types here. To send APM traces to this Source, configure your Datadog tracing library (`dd-trace`) to send traces directly to Cribl Edge instead of via the Datadog Agent.	`DD_APM_DD_URL`

Managing API Keys

Suppose your data flow runs from Datadog Agents, to Cribl Edge Datadog Agent Sources, to Cribl Edge Datadog Destinations, to Datadog accounts. You'll need to decide **how many** of each of these elements there are to define the data flow you want. You will also set (or override) **Datadog API keys** to support the desired data flow. For some data flows, you'll need the **General Settings** > **Allow API key from events** toggle in the Cribl Edge Datadog Destination.

Another possibility is that you want data to flow to some Destination other than a Cribl Edge Datadog Destination. If that's the case, try adapting the examples below to your use case. Please connect with us on the `Cribl Community` Slack if you have questions.

Data Flow Examples

The examples which follow start simple and become more complex in terms of desired data flow, and, consequently, of Cribl Edge configuration.

One Agent to One Account

- You're running one Datadog Agent on one host.
- You want the output of the Agent sent to a single Datadog account.
- You only need one API key.
- You configure that single API key on your Cribl Edge Datadog Destination.

Many Agents to One Account

- You're running Datadog Agents on a fleet of hosts.
- You want to consolidate all the output of the Agents into a single Datadog account.
- You only need one API key. This is no different from the simpler one-to-one scenario above.

- You configure that single API key on your Cribl Edge Datadog Destination.
- No matter which host the data originates from, your Cribl Edge Datadog Destination sends it to Datadog using that single API key.

Many Agents to Many Accounts

- You're running Datadog Agents on a fleet of hosts.
- You want the output of the Agents from some hosts to flow into one Datadog account, and the output from other hosts to flow into a different Datadog account.
    - This may need to scale up to multiple accounts, if, for example, you are managing data from several different customers, or from several different organizations within your company.
- You need one API key **for each** Datadog account.
- In the Cribl Edge Datadog Destination, toggle **General Settings** > **Allow API key from events** on.
- Here's what happens:
    - Datadog Agent always sends an API key when it communicates with the Cribl Edge Datadog Agent Source. Agents within different subgroups of hosts will send different API keys, as described above.
    - Cribl Edge Datadog Agent Source passes the API key from the Agent to the Cribl Edge Datadog Destination as an internal field.
    - Cribl Edge Datadog Destination uses that API key to direct its output to the correct Datadog account. If need be, you can configure the Destination to override the passed-in API key with a different one. This comes in handy when you know that your Agent(s) are using invalid API keys. You can even override **all** passed-in API keys.

Verifying that Data is Flowing

Once you have configured your Datadog Agent(s) as described above, and they have begun sending data:

- Try viewing the incoming data in the **Live Data** tab in your Cribl Edge Datadog Agent Source.
- If you have a Cribl Edge Datadog Destination set up, connect your Cribl Edge Datadog Agent Source to it via a **QuickConnect Passthru**, and verify that the same data shows up in the Destination's **Live Data** tab.
    - Another possibility is that you want your Cribl Edge Datadog Agent Source to connect to some Destination other than a Cribl Edge Datadog Destination. That's beyond the scope of this example, but verifying data flow should be similar.
- If the Cribl Edge Datadog Destination is configured to send data to a Datadog instance, verify that the same data shows up there, in the appropriate form, according to the transformations you've configured in Cribl Edge.
- If you have a many-to-many data flow as described above, verify that output is being correctly split among the possible Datadog accounts at the end of the data flow.

Assuming that data is behaving as expected, you can proceed to the best part, namely adding one or more Pipelines between Datadog Agent Source and Datadog Destination, to transform the data however you wish. Given the varied nature of what Datadog Agents collect, there should be ample opportunities to make the data leaner and more usable.

Troubleshooting

The Source's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they are ingested. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the ingestion process, including any errors or warnings that may have occurred.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify ingestion issues. Analyze the graphs showing events and bytes in/out over time.

Common Issue

Dropping request because token invalid",“authToken”: “Bas...Njc=”

The specified token is invalid. Note that the above message is logged only at the debug level.

12.2.10. Elasticsearch API Source

Cribl Edge supports receiving data over HTTP/S using the Elasticsearch Bulk API.

Type: **Push** | TLS Support: **YES** | Event Breaker Support: **No**

This Source supports gzip-compressed inbound data when the `Content-Encoding: gzip` connection header is set.

For examples of receiving data from popular senders via this API, see Configuring Elastic Beats below.

Looking for the Elasticsearch API Source in **Cribl Search**? See Ingest Elasticsearch Data into Cribl Search.

Configure Cribl Edge to Receive Elasticsearch Bulk API Data over HTTP(S)

Cribl Edge ships with an Elasticsearch API Source preconfigured to listen on Port 9200. You can clone or directly modify this Source to further configure it, and then enable it.

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Source** and select the Source you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). Select the Source you want. Next, select **Add Source**.
2. In the **New Source** modal, configure the following under **General Settings**:
    - **Enabled**: Toggle on to enable the Source.
    - **Input ID**: Enter a unique name to identify this Syslog Source definition. The default Source is prefilled with the value `in_elastic`. which can't be changed via the UI. If you clone this Source, Cribl Edge will add `-CLONE` to the original **Input ID**.
    - **Description**: Optionally, enter a description.
    - **Address**: Enter the hostname/IP on which to listen for Elasticsearch data. (For example, `localhost` or `0.0.0.0`.)
    - **Port**: Enter the port number to listen on.

- **Elasticsearch API endpoint** (for Bulk API): Absolute path on which to listen for Elasticsearch API requests. Defaults to `/`. Cribl Edge automatically appends `_bulk`, so (for example) `/myPath` becomes `/myPath/_bulk`. Requests could then be made to either `/myPath/_bulk` or `/myPath/<myIndexName>/_bulk`. Other entries are faked as success.

3. Next, you can configure the following **Optional Settings**:
   - **Tags**: Optionally, add tags that you can use to filter and group Sources in Cribl Edge's UI. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
4. Optionally, you can adjust the Authentication, TLS, Persistent Queue Settings, Processing, and Advanced settings, or Connected Destinations outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

Authentication

In the **Authentication type** drop-down, select one of the following options:

- **None**: Don't use authentication.
- **Basic**: Displays **Username** and **Password** fields for you to enter HTTP Basic authentication credentials. Click **Generate** if you need a new password.
- **Basic (credentials secret)**: Provide username and password credentials referenced by a secret. Select a stored text secret in the resulting **Credentials secret** drop-down, or select **Create** to configure a new secret.
- **Auth tokens**: Use HTTP token authentication. Click **Add Token** and, in the resulting **Token** field, enter the bearer token that must be included in the HTTP authorization header. Click **Generate** if you need a new token. Click **Add Token** to display additional rows to specify more tokens.

See also Periodic Logging for information on how auth tokens affect product logging.

TLS Settings (Server Side)



On Cribl.Cloud, you can enable TLS using Cribl-provided certficates if desired.

Enabled: Defaults to toggled off. When toggled on:

Certificate: Name of the predefined certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

Passphrase: Passphrase to use to decrypt private key.

Certificate path: Server path containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

CA certificate path: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

Authenticate client (mutual auth): Require clients to present their certificates. Used to perform mutual authentication using SSL certs. Default is toggled off. When toggled on:

- **Validate client certificates**: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (for example, the system's CA). Default is toggled on.
- **Common name**: Regex that a peer certificate's subject attribute must match in order to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. (For example, to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.) If the subject attribute contains Subject Alternative Name (SAN) entries, the Source will check the regex against all of those but ignore the Common Name (CN) entry (if any). If the certificate has no SAN extension, the Source will check the regex against the single name in the CN.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In the **Persistent Queue Settings** tab, you can optionally specify persistent queue storage, using the following controls. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Source Persistent Queues (sPQ)

> On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the **Enable persistent queue** toggle. If enabled, PQ is automatically configured in `Always On` mode, with a maximum queue size of 1 GB disk space allocated per PQ-enabled Source, per Worker Process.
>
> The 1 GB limit is on uncompressed inbound data, and the queue does not perform any compression. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable persistent queue: Default is toggled off. When toggled on:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to the Cribl Stream data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from the Cribl Stream data processing engine.



`Smart` mode only engages when necessary, such as when a downstream Destination becomes blocked *and* the **Buffer size limit** reaches its limit. When persistent queue is set to `Smart` mode, Cribl attempts to flush the queue when every new event arrives. The only time events stay in the buffer is when a downstream Destination becomes blocked.

Buffer size limit: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. This buffer is for all connections, not just per Worker Process. For that reason, this can dramatically expand memory usage. Connections share this limit, which may result in slightly lower throughput for higher numbers of connections. For higher numbers of connections, consider increasing the limit.

Commit frequency: The number of events to send downstream before committing that Stream has read them. Defaults to `42`.

File size limit: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, and so forth. If not specified, Cribl Stream applies the default `1 MB`.

Queue size limit: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data. Required, and defaults to `5 GB`. Accepts positive numbers with units of `KB`, `MB`, `GB`, and so forth. Can be set as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** in Group or Fleet settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file closes. Defaults to `None`; `Gzip` is also available.



In Cribl Stream 4.1 and newer, the Source persistent queue default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Optimize Source Persistent Queues (sPQ).

You can optimize Workers' startup connections and CPU load at **Group/Fleet settings** > Worker Processes.

Processing Settings Fields

In this section, you can define new fields or modify existing ones using JavaScript expressions, similar to the Eval function.

- The **Field Name** can either be a new field (unique within the event) or an existing field name to modify its value.
- The **Value** is a JavaScript expression (enclosed in quotes or backticks) to compute the field's value (can be a constant). Select this field's advanced mode icon (far right) if you'd like to open a modal where you can work with sample data and iterate on results.

This flexibility means you can:

- Add new fields to enrich the event.
- Modify existing fields by overwriting their values.
- Compute logic or transformations using JavaScript expressions.

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline or Pack to process data from this input before the data is sent through the Routes.

Advanced Settings

Show originating IP: Toggle on when clients are connecting through a proxy that supports the `X-Forwarded-For` header to keep the client's original IP address on the event instead of the proxy's IP address. This setting affects how the Source handles the `__srcIpPort` field.

Capture request headers: Toggle on to add request headers to events, in the `__headers` field.

Health check endpoint: Toggle on to enable a health check endpoint specific to this Source, `http(s)://<host>:<port>/cribl_health`. A `200` HTTP response code is returned when the Source is healthy. Otherwise, two errors you could receive are:

- `ECONNRESET` where the Source failed to initialize due to not having listeners on the port.
- `503` or `Server is busy, max active connections reached` indicate there are too many connections per Worker Process.

Active request limit (on-prem only): Maximum number of active requests allowed for this Source, per Worker Process. Defaults to `256`. Enter `0` for unlimited.



Raising this limit can increase throughput by allowing more concurrent data requests, but increases resource usage and load on both your Cribl Edge infrastructure and on downstream Destinations. Before raising this limit, ensure:

- Your Cribl Edge deployment has sufficient capacity to support higher request concurrency, including CPU, memory, or number of Worker Processes.
- Downstream Destinations are correctly sized and tuned to accept the higher data ingest rate, preventing backpressure. See Manage Backpressure for more information.

Improper sizing on either side can result in dropped events, delayed processing, or overall system instability.

Activity log sample rate: Determines how often request activity is logged at the `info` level. The default `100` value logs every 100th value; a `1` value would log every request; a `10` value would log every 10th request; etc.

Requests-per-socket limit: The maximum number of requests Cribl Edge should allow on one socket before instructing the client to close the connection. Defaults to `0` (unlimited). See Balancing Connection Reuse Against Request Distribution below.

Socket timeout (seconds): How long Cribl Edge should wait before assuming that an inactive socket has timed out. The default `0` value means wait forever.

Request timeout (seconds): How long to wait for an incoming request to complete before aborting it. The default `0` value means wait indefinitely.

Keep-alive timeout (seconds): After the last response is sent, Cribl Edge will wait this long for additional data before closing the socket connection. Defaults to `5` seconds; minimum is `1` second; maximum is `600` seconds (10 minutes).

The longer the **Keep-alive timeout**, the more Cribl Edge will reuse connections. The shorter the timeout, the closer Cribl Edge gets to creating a new connection for every request. When request frequency is high, you can use longer timeouts to reduce the number of connections created, which mitigates the associated cost.

IP allowlist regex: Grants access to requests originating from specific IP addresses that match a defined pattern. Unmatched requests are rejected with a 403 (Forbidden) status code. Defaults to `.*` (allow all).

IP denylist regex: Blocks requests originating from specific IP addresses that match a defined pattern, even if they would be allowed by default. Rejected requests receive a 403 (Forbidden) status code. Defaults to `^$` (allow all).

Enable proxy mode: If toggled on, see Proxy Mode below for the resulting options.

Extra HTTP headers: Name-value pairs to pass as additional HTTP headers. By default, Cribl Edge's responses to HTTP requests include the `X-elastic-product` header, with an `Elasticsearch` value. (This is required by certain clients, including some Elastic Beats.)

API version: To upstream Elastic Beats, this Cribl Edge Source will appear as an Elasticsearch instance matching the version that you set in this drop-down:

- `6.8.4` - Retained for backward compatibility.
- `8.3.2` - This default entry matches Elasticsearch's current `8.3.x` versions.
- `Custom` - Opens an HTTP response object in the **Custom API Version** editor. This object replicates what an Elasticsearch server would send in its HTTP responses to a client. You can edit the `version : number`, and any other fields, as required to satisfy the Elastic Beat sending data to this Source. (This `Custom` option supports future Elasticsearch releases, as long as Elasticsearch keeps the same response-object structure.)

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Proxy Mode

Enabling proxy mode allows Cribl Edge to proxy non-Bulk API requests to a downstream Elasticsearch server. This can be useful when integrating with Elasticsearch API senders like Elastic Endgame agents, which send requests that Cribl Edge does not natively support. Toggle **Enable proxy mode** on to expose the following controls.

Proxy URL: URL of the Elasticsearch server that will proxy non-bulk requests, for example: `http://elastic:9200`.

Reject unauthorized certificates: Reject certificates that are not authorized by a trusted CA (for example, the system's CA). Default is toggled on.

Remove headers: Enter any headers that you want removed from proxied requests. Press `Tab` or `Return` to separate header names.

Proxy request timeout: How long, in seconds, to wait for a proxy request to complete before aborting it. Defaults to `60` seconds; minimum timeout is `1` second.

To understand how **Proxy request timeout** interacts with the `X-Forwarded-For` header, see Overriding `__srcIpPort` with Client IP/Port.

Authentication method: Select one of the following options.

- **None**: Don't use authentication.
- **Manual**: Displays **Username** and **Password** fields for you to enter HTTP Basic authentication credentials.
- **Secret**: This option exposes a **Credentials secret** drop-down, in which you can select a stored secret that references the credentials described above. A **Create** link is available to store a new, reusable secret.

Balancing Connection Reuse Against Request Distribution

Requests-per-socket limit allows you to limit the number of HTTP requests an upstream client can send on one network connection. Once the limit is reached, Cribl Edge uses HTTP headers to inform the client that it must establish a new connection to send any more requests. (Specifically, Cribl Edge sets the HTTP `Connection` header to `close`.) After that, if the client disregards what Cribl Edge has asked it to do and tries to send another HTTP request over the existing connection, Cribl Edge will respond with an HTTP status code of `503 Service Unavailable`.

Use this setting to strike a balance between connection reuse by the client, and distribution of requests among one or more Edge Node processes by Cribl Edge:

- When a client sends a sequence of requests on the same connection, that is called connection reuse. Because connection reuse benefits client performance by avoiding the overhead of creating new connections, clients have an incentive to **maximize** connection reuse.
- Meanwhile, a single process on that Edge Node will handle all the requests of a single network connection, for the lifetime of the connection. When receiving a large overall set of data, Cribl Edge performs better when the workload is distributed across multiple Edge Node processes. In that situation, it makes sense to **limit** connection reuse.

There is no one-size-fits-all solution, because of variation in the size of the payload a client sends with a request and in the number of requests a client wants to send in one sequence. Start by estimating how long connections will stay open. To do this, multiply the typical time that requests take to process (based on payload size) times the number of requests the client typically wants to send.

If the result is 60 seconds or longer, set **Requests-per-socket limit** to force the client to create a new connection sooner. This way, more data can be spread over more Edge Node processes within a given unit of time.

For example: Suppose a client tries to send thousands of requests over a very few connections that stay open for hours on end. By setting a relatively low **Requests-per-socket limit**, you can ensure that the same work is done over more, shorter-lived connections distributed between more Edge Node processes, yielding better performance from Cribl Edge.

A final point to consider is that one Cribl Edge Source can receive requests from more than one client, making it more complicated to determine an optimal value for **Requests-per-socket limit**.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

Field Normalization

When ingesting data from Elasticsearch, this Source performs the following field normalization:

- Renames `@timestamp` to `_time`, maintaining millisecond precision.
- Renames `host.name` to `host`.
- Stores original `host.name` in `__host` to retain all host-related information.

The Elasticsearch Destination does the reverse, and it also recognizes the presence of `__host`.

Internal Fields

Cribl Edge uses a set of internal fields to assist in handling of data. These "meta" fields are **not** part of an event, but they are accessible, and Functions can use them to make processing decisions.

Fields for this Source:

- `__headers` - Added only when **Advanced Settings** > **Capture request headers** is toggled on.
- `__host`
- `__id`
- `__index`
- `__inputId`
- `__pipeline` - If present in the Elasticsearch `event.action` field of the event. See also how to override the pipeline attribute.
- `__srcIpPort` - See details below.
- `__type`

Overriding the Pipeline Attribute in the Elasticsearch and Elastic Cloud Destinations

The Elasticsearch Source will record the pipeline attribute received in a variable named `__pipeline`, if pipeline was presented in the Source event. If you want to forward the pipeline from the Source event to an Elasticsearch or Elastic Cloud Destination, set up the **Elastic pipeline** in one of the following ways.

This expression will use the value of `__pipeline` or default to `'myPipeline'` if `__pipeline` is missing:

```
__pipeline || 'myPipeline'
```

An alternative is to pass the pipeline if present, or otherwise omit it:

```
__pipeline ? __pipeline : undefined
```

Overriding `__srcIpPort` with Client IP/Port

The `__srcIpPort` field's value contains the IP address and (optionally) port of the Elasticsearch client sending data to this Source.

When any proxies (including load balancers) lie between the Elasticsearch client and the Source, the last proxy adds an `X-Forwarded-For` header whose value is the IP/port of the original client. With multiple proxies, this header's value will be an array, whose first item is the original client IP/port.

If `X-Forwarded-For` is present, and **Advanced Settings** > **Show originating IP** is toggled off, the original client IP/port in this header will override the value of `__srcIpPort`.

If **Show originating IP** is toggled on, the `X-Forwarded-For` header's contents will **not** override the `__srcIpPort` value. (Here, the upstream proxy can convey the client IP/port without using this header.)

Configuring Elastic Beats

Beats are open-source data shippers that act as agents, sending data to Elasticsearch (or to other services, in this case Cribl Edge). The Beats most popular with Cribl users are Filebeat and Winlogbeat.

To set up a Beat to send data to Cribl Edge, edit the Beat's YAML configuration file: `filebeat.yml` for Filebeat, `winlogbeat.yml` for Winlogbeat, and so on. In the config file, you'll specify your Cribl Edge Elasticsearch Source endpoint as the Beat's Elasticsearch output. To the Beat, Cribl Edge will appear as an instance of Elasticsearch.

If you're using HTTP token authentication (which is disabled by default, both on-prem and on Cribl.Cloud): First, set the token. Then add the following to the Beat config file under `output.elasticsearch.headers`, substituting your token for `myToken42`:

```
output.elasticsearch:
  headers:
    Authorization: "myToken42"
```

Copy

Configuring an Elastic Agent

An Elastic Agent is single agent for logs, metrics, security data, and threat prevention that sends data to Elasticsearch (or to other services, in this case Cribl Edge).

When you are sending from an Elastic Agent to Cribl Edge, the `elastic-agent.yml` file doesn't pay any attention to the settings for `output.elasticsarch.allow_older_versions: true`. As a result, the Elasticsearch API Source will not get any data.

To set it up correctly, go to the **Advanced Settings** tab, and change the **API Version** to `Custom`. In the **Custom API Version** editor, edit the `version : number` to match the version of the Elastic Agent you are using. This should allow the data to start flowing to the Source.

Periodic Logging

Cribl Edge logs metrics about incoming requests and ingested events once per minute.

If one or more auth tokens are configured and enabled, Cribl Edge logs requests and events for each enabled auth token individually. Since the tokens themselves are redacted for security, Cribl Edge logs the initial text of the token description to help you identify which token a given log is for.

If no auth token is configured and enabled, Cribl Edge simply logs overall statistics about incoming requests and ingested events.

These logs are stored in the `metrics.log` file. To view them in the UI, open the Source's **Logs** tab and choose **Worker Process X Metrics** from the drop-down, where **X** is the desired Worker process.

Troubleshooting

The Source's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they are ingested. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the ingestion process, including any errors or warnings that may have occurred.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify ingestion issues. Analyze the graphs showing events and bytes in/out over time.

Common Issue

Dropping request because token invalid",“authToken”: “Bas...Njc=”

The specified token is invalid. Note that the above message is logged only at the debug level.

12.2.11. HTTP/S (Bulk API) Source

Cribl Edge supports receiving data over HTTP/S from Cribl Bulk API, Splunk HEC, and Elastic Bulk API endpoints.

Type: **Push** | TLS Support: **YES** | Event Breaker Support: **No**

This Source supports gzip-compressed inbound data when the `Content-Encoding: gzip` connection header is set.

Configure Cribl Edge to Receive Data over HTTP(S)

Cribl Edge ships with an HTTP Source preconfigured to listen on Port 10080, and on several default endpoints. You can clone or directly modify this Source to further configure it, and then enable it.

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Source** and select the Source you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). Select the Source you want. Next, select **Add Source**.
2. In the **New Source** modal, configure the following under **General Settings**:
    - **Input ID**: Enter a unique name. The default Source is prefilled with the value `http`, which can't be changed via the UI. If you clone this Source, Cribl Edge will add `-CLONE` to the original **Input ID**.
    - **Description**: Optionally, enter a description.
    - **Address**: Enter the hostname/IP on which to listen for HTTP(S) data. (For example, `localhost` or `0.0.0.0`.)
    - **Port**: Enter the port number to listen on.
3. Under **Authentication**, enter the **Auth tokens**:
    - **Token**: Shared secrets to be provided by any client (`Authorization: <token>`). Click **Generate** to create a new secret. If empty, unauthorized access **will be permitted**.
    - **Description**: Optionally, enter a description.

- **Fields**: Fields to add to events referencing this token. Each field is a **Name/Value** pair, where **Value** is a JavaScript expression to compute field's value, enclosed in quotes or backticks. (Can evaluate to a constant.) For details, see Authentication Fields.

4. Next, you can configure the following **Optional Settings**:
   - **Cribl HTTP event API**: Base path on which to listen for Cribl HTTP API requests. To construct the actual endpoint, Cribl Edge will append `/_bulk` to this path. For example, with the default value of `/cribl`, your senders should send events to a `/cribl/_bulk` path. Use an empty string to disable.
   - **Elastic API endpoint** (for Bulk API): Base path on which to listen for Elasticsearch API requests. For details, see Elastic API Endpiont.
   - **Splunk HEC endpoint**: Absolute path on which to listen for Splunk HTTP Event Collector (HEC) API requests. Use an empty string to disable. Default entry is `/services/collector`. For details, see Splunk HEC Endpoint.
   - **Enable Splunk HEC acknowledgements**: Whether to enable Splunk HEC acknowledgements. Default is toggled off.
   - **Tags**: Optionally, add tags that you can use to filter and group Sources in Cribl Edge's UI. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
5. Optionally, you can adjust the TLS, Persistent Queue Settings, Processing and Advanced settings, or Connected Destinations outlined in the sections below.
6. Select **Save**, then **Commit & Deploy**.

Elastic API Endpoint

Currently, the only supported option is the default `/elastic`, to which Cribl Edge will append `/_bulk`. So, your senders should send events to an `/elastic/_bulk` path. Other entries are faked as success. Use an empty string to disable.

Cribl generally recommends that you use the dedicated Elasticsearch API Source instead of this endpoint. The Elastic API implementation here is provided for backward compatibility, and for users who want to ingest multiple inputs on one HTTP/S port.

Splunk HEC Endpoint

This Splunk HEC implementation is an **event** (such as, not **raw**) endpoint. For details, see Splunk's documentation. To send data to it from a HEC client, use either `/services/collector` or `/services/collector/event`. (See the examples below.)

The system default Event Breaker is automatically applied, which enforces a limit of 51,200 bytes per event.

Cribl generally recommends that you use the dedicated Splunk HEC Source instead of this endpoint. The Splunk HEC implementation here is provided for backward compatibility, and for users who want to ingest multiple inputs on one HTTP/S port.

Authentication Fields

By default, fields specified here will override fields of the same name in incoming events. However, you can customize this behavior to allow event fields to take precedence. For example, to allow the event's `index` field to take precedence, you could use this expression: `` `event.index || 'defaultIndex'` ``.

Fields add context to events based on the token used. This can be useful for tagging events with metadata such as source or environment, enhancing security by avoiding direct use of authorization values in Routes and Pipelines.

See also Periodic Logging for information on how auth tokens affect product logging.

TLS Settings (Server Side)



On Cribl.Cloud, you can enable TLS using Cribl-provided certficates if desired.

Enabled: Defaults to toggled off. When toggled on:

Certificate: Name of the predefined certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

Passphrase: Passphrase to use to decrypt private key.

Certificate path: Server path containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

CA certificate path: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

Authenticate client (mutual auth): Require clients to present their certificates. Used to perform mutual authentication using SSL certs. Default is toggled off. When toggled on:

- **Validate client certificates**: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (for example, the system's CA). Default is toggled on.
- **Common name**: Regex that a peer certificate's subject attribute must match in order to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. (For example, to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.) If the subject attribute contains Subject Alternative Name (SAN) entries, the

Source will check the regex against all of those but ignore the Common Name (CN) entry (if any). If the certificate has no SAN extension, the Source will check the regex against the single name in the CN.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In the **Persistent Queue Settings** tab, you can optionally specify persistent queue storage, using the following controls. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Source Persistent Queues (sPQ)

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the **Enable persistent queue** toggle. If enabled, PQ is automatically configured in `Always On` mode, with a maximum queue size of 1 GB disk space allocated per PQ-enabled Source, per Worker Process.

The 1 GB limit is on uncompressed inbound data, and the queue does not perform any compression. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable persistent queue: Default is toggled off. When toggled on:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to the Cribl Stream data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from the Cribl Stream data processing engine.

`Smart` mode only engages when necessary, such as when a downstream Destination becomes blocked *and* the **Buffer size limit** reaches its limit. When persistent queue is set to `Smart` mode, Cribl attempts to flush the queue when every new event arrives. The only time events stay in the buffer is when a downstream Destination becomes blocked.

Buffer size limit: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. This buffer is for all connections, not just per Worker Process. For that reason, this can dramatically expand memory usage. Connections share this limit, which may result in slightly lower throughput for higher numbers of connections. For higher numbers of connections, consider increasing the limit.

Commit frequency: The number of events to send downstream before committing that Stream has read them. Defaults to `42`.

File size limit: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, and so forth. If not specified, Cribl Stream applies the default `1 MB`.

Queue size limit: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, and so forth. Can be set as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** in Group or Fleet settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file closes. Defaults to `None`; `Gzip` is also available.



In Cribl Stream 4.1 and newer, the Source persistent queue default **Mode** is `Always On`, to best ensure events' delivery. For details on optimizing this selection, see Optimize Source Persistent Queues (sPQ).

You can optimize Workers' startup connections and CPU load at **Group/Fleet settings** > Worker Processes.

Processing Settings Fields

In this section, you can define new fields or modify existing ones using JavaScript expressions, similar to the Eval function.

- The **Field Name** can either be a new field (unique within the event) or an existing field name to modify its value.
- The **Value** is a JavaScript expression (enclosed in quotes or backticks) to compute the field's value (can be a constant). Select this field's advanced mode icon (far right) if you'd like to open a modal where you can work with sample data and iterate on results.

This flexibility means you can:

- Add new fields to enrich the event.

- Modify existing fields by overwriting their values.
- Compute logic or transformations using JavaScript expressions.

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline or Pack to process data from this input before the data is sent through the Routes.

Advanced Settings

Show originating IP: Toggle on when clients are connecting through a proxy that supports the `X-Forwarded-For` header to keep the client's original IP address on the event instead of the proxy's IP address. This setting affects how the Source handles the `__srcIpPort` field.

Capture request headers: Toggle on to add request headers to events, in the `__headers` field.

Health check endpoint: Toggle on to enable a health check endpoint specific to this Source, `http(s)://<host>:<port>/cribl_health`. A `200` HTTP response code is returned when the Source is healthy. Otherwise, two errors you could receive are:

- `ECONNRESET` where the Source failed to initialize due to not having listeners on the port.
- `503` or `Server is busy, max active connections reached` indicate there are too many connections per Worker Process.

Active request limit (on-prem only): Maximum number of active requests allowed for this Source, per Worker Process. Defaults to `256`. Enter `0` for unlimited.



Raising this limit can increase throughput by allowing more concurrent data requests, but increases resource usage and load on both your Cribl Edge infrastructure and on downstream Destinations. Before raising this limit, ensure:

- Your Cribl Edge deployment has sufficient capacity to support higher request concurrency, including CPU, memory, or number of Worker Processes.
- Downstream Destinations are correctly sized and tuned to accept the higher data ingest rate, preventing backpressure. See Manage Backpressure for more information.

Improper sizing on either side can result in dropped events, delayed processing, or overall system instability.

Activity log sample rate: Determines how often request activity is logged at the `info` level. The default `100` value logs every 100th value; a `1` value would log every request; a `10` value would log every 10th request; and so forth.

Requests-per-socket limit: The maximum number of requests Cribl Edge should allow on one socket before instructing the client to close the connection. Defaults to `0` (unlimited). See Balancing Connection Reuse Against Request Distribution below.

Socket timeout (seconds): How long Cribl Edge should wait before assuming that an inactive socket has timed out. The default `0` value means wait forever.

Request timeout (seconds): How long to wait for an incoming request to complete before aborting it. The default `0` value means wait indefinitely.

Keep-alive timeout (seconds): After the last response is sent, Cribl Edge will wait this long for additional data before closing the socket connection. Defaults to `5` seconds; minimum is `1` second; maximum is `600` seconds (10 minutes).



The longer the **Keep-alive timeout**, the more Cribl Edge will reuse connections. The shorter the timeout, the closer Cribl Edge gets to creating a new connection for every request. When request frequency is high, you can use longer timeouts to reduce the number of connections created, which mitigates the associated cost.

IP allowlist regex: Grants access to requests originating from specific IP addresses that match a defined pattern. Unmatched requests are rejected with a 403 (Forbidden) status code. Defaults to `.*` (allow all).

IP denylist regex: Blocks requests originating from specific IP addresses that match a defined pattern, even if they would be allowed by default. Rejected requests receive a 403 (Forbidden) status code. Defaults to `^$` (allow all).

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Balancing Connection Reuse Against Request Distribution

Requests-per-socket limit allows you to limit the number of HTTP requests an upstream client can send on one network connection. Once the limit is reached, Cribl Edge uses HTTP headers to inform the client that it must establish a new connection to send any more requests. (Specifically, Cribl Edge sets the HTTP `Connection` header to `close`.) After that, if the client disregards what Cribl Edge has asked it to do and tries to send another HTTP request over the existing connection, Cribl Edge will respond with an HTTP status code of `503 Service Unavailable`.

Use this setting to strike a balance between connection reuse by the client, and distribution of requests among one or more Edge Node processes by Cribl Edge:

- When a client sends a sequence of requests on the same connection, that is called connection reuse. Because connection reuse benefits client performance by avoiding the overhead of creating new

connections, clients have an incentive to **maximize** connection reuse.

- Meanwhile, a single process on that Edge Node will handle all the requests of a single network connection, for the lifetime of the connection. When receiving a large overall set of data, Cribl Edge performs better when the workload is distributed across multiple Edge Node processes. In that situation, it makes sense to **limit** connection reuse.

There is no one-size-fits-all solution, because of variation in the size of the payload a client sends with a request and in the number of requests a client wants to send in one sequence. Start by estimating how long connections will stay open. To do this, multiply the typical time that requests take to process (based on payload size) times the number of requests the client typically wants to send.

If the result is 60 seconds or longer, set **Requests-per-socket limit** to force the client to create a new connection sooner. This way, more data can be spread over more Edge Node processes within a given unit of time.

For example: Suppose a client tries to send thousands of requests over a very few connections that stay open for hours on end. By setting a relatively low **Requests-per-socket limit**, you can ensure that the same work is done over more, shorter-lived connections distributed between more Edge Node processes, yielding better performance from Cribl Edge.

A final point to consider is that one Cribl Edge Source can receive requests from more than one client, making it more complicated to determine an optimal value for **Requests-per-socket limit**.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

Internal Fields

Cribl Edge uses a set of internal fields to assist in handling of data. These "meta" fields are **not** part of an event, but they are accessible, and Functions can use them to make processing decisions.

Fields for this Source:

- `__headers` - Added only when **Advanced Settings** > **Capture request headers** is toggled on.
- `__inputId`
- `__hecToken`
- `__srcIpPort` - See details below.

- `__host` (Elastic In)
- `__id` (Elastic In)
- `__index` (Elastic In)
- `__type` (Elastic In)

Overriding `__srcIpPort` with Client IP/Port

The `__srcIpPort` field's value contains the IP address and (optionally) port of the client sending data to this Source.

When any proxies (including load balancers) lie between the HTTP client and the Source, the last proxy adds an `X-Forwarded-For` header whose value is the IP/port of the original HTTP client. With multiple proxies, this header's value will be an array, whose first item is the original client IP/port.

If `X-Forwarded-For` is present, and **Advanced Settings** > **Show originating IP** is toggled off, the original client IP/port in this header will override the value of `__srcIpPort`.

If **Show originating IP** is toggled on, the `X-Forwarded-For` header's contents will **not** override the `__srcIpPort` value. (Here, the upstream proxy can convey the client IP/port without using this header.)

Format and Endpoint

Cribl Edge expects HTTP(S) events to be formatted as one JSON record per event. Here are two event records:

Sample Event Format

```
{"_time":1541280341, "_raw":"this is a sample event ", "host":"myHost", "source":"myS
{"_time":1541280341, "host":"myOtherHost", "source":"myOtherSource", "_raw": "{\"mess
```

Copy

Note 1: Events can be sent as separate POSTs, but Cribl **highly** recommends combining multiple events in newline-delimited groups, and POSTing them together.

Note 2: If an HTTP(S) source is routed to a Splunk destination, fields within the JSON payload are mapped to Splunk fields. Fields that do not have corresponding (native) Splunk fields become index-time fields. For example, let's assume we have a HTTP(S) event like this:

```
{"_time":1541280341, "host":"myHost", "source":"mySource", "_raw":"this is a sample event ", "fieldA":"valueA"}
```

Here, `_time`, `host` and `source` become their corresponding fields in Splunk. The value of `_raw` becomes the actual body of the event, and `fieldA` becomes an index-time field. (`fieldA::valueA`).

Examples

The examples in this section demonstrate sending HTTP data into a Bulk HTTP Source configured to listen on port `10080`. The token is set to `myToken42` and included in the `Authorization` header of requests.

Cribl Endpoint - Single Event

Cribl Endpoint, Single Event Example

```
curl -k "http://<myCriblHost>:10080/cribl/_bulk" \
-H "Authorization: myToken42" \
-d '{"_raw":"this is a sample event ", "host":"myHost", "source":"mySource", "fieldA'
```

Copy

Cribl Endpoint - Multiple Events

Cribl Endpoint, Multiple Events Example

```
curl -k "http://<myCriblHost>:10080/cribl/_bulk" \
-H "Authorization: myToken42" \
-d $'{"_raw":"this is a sample event ", "host":"myHost", "source":"mySource", "fieldA
{"_raw":"this is another sample event ", "host":"myOtherHost", "source":"myOtherSourc
```

Copy

Splunk HEC Event Endpoints

For Splunk HEC, the token specification can be either `Splunk <token>` or `<token>`.

Splunk HEC Event Endpoint, /services/collector/event Example

```
curl -k "http://<myCriblHost>:10080/services/collector/event" \
-H "Authorization: myToken42" \
-d '{"event":"this is a sample event ", "host":"myHost", "source":"mySource", "fieldA
```

Copy

Splunk HEC Event Endpoint, /services/collector Example

```
curl -k "http://<myCriblHost>:10080/services/collector" \
-H "Authorization: myToken42" \
-d '{"event":"this is a sample event ", "host":"myHost", "source":"mySource", "fieldA
```

Copy

Cribl.Cloud - Single Event

Replace `<organizationId>` with your Cribl.Cloud Organization ID.

Cribl.Cloud, Single Event Example

```
curl -k "https://default.main.<organizationId>.cribl.cloud:10080/cribl/_bulk" \
-H "Authorization: myToken42" \
-d '{"_raw":"this is a sample event ", "host":"myHost", "source":"mySource", "fieldA'
```

Copy

For Cribl.Cloud Enterprise plans, replace `default` in the request URL with the name of the Fleet that you want to send to.

Periodic Logging

Cribl Edge logs metrics about incoming requests and ingested events once per minute.

If one or more auth tokens are configured and enabled, Cribl Edge logs requests and events for each enabled auth token individually. Since the tokens themselves are redacted for security, Cribl Edge logs the initial text of the token description to help you identify which token a given log is for.

If no auth token is configured and enabled, Cribl Edge simply logs overall statistics about incoming requests and ingested events.

These logs are stored in the `metrics.log` file. To view them in the UI, open the Source's **Logs** tab and choose **Worker Process X Metrics** from the drop-down, where **X** is the desired Worker process.

Troubleshooting

The Source's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they are ingested. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the ingestion process, including any errors or warnings that may have occurred.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify ingestion issues. Analyze the graphs showing events and bytes in/out over time.

Common Issue

Dropping request because token invalid",“authToken”: “Bas...Njc=”

The specified token is invalid. Note that the above message is logged only at the debug level.

12.2.12. Raw HTTP/S Source

Cribl Edge supports receiving raw HTTP data. The Raw HTTP Source listens on a specific port, captures every HTTP request to that port, and creates a corresponding event that it pushes to its configured Event Breakers.



Type: **Push** | TLS Support: **YES** | Event Breaker Support: **YES**

This Source supports gzip-compressed inbound data when the `Content-Encoding: gzip` connection header is set.

Looking for the Raw HTTP Source in **Cribl Search**? See Ingest Raw HTTP Data into Cribl Search.

Configure Cribl Edge to Receive Raw HTTP Data

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Source** and select the Source you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). Select the Source you want. Next, select **Add Source**.
2. In the **New Source** modal, configure the following under **General Settings**:
    - **Input ID**: Enter a unique name to identify this Raw HTTP Source definition. If you clone this Source, Cribl Edge will add `-CLONE` to the original **Input ID**.
    - **Description**: Optionally, enter a description.
    - **Address**: Enter the address to bind on. Defaults to `0.0.0.0` (all addresses).
    - **Port**: Enter the port number to listen on. For Cribl.Cloud, use the `20000 - 20010` port range.
3. Under **Authentication**, enter the **Auth tokens**:
    - **Token**: Shared secrets to be provided by any client. Click **Generate** to create a new secret. If empty, permits open access.
    - **Description**: Optionally, enter a description.
    - **Fields**: Fields to add to events referencing this token. Each field is a **Name/Value** pair, where **Value** is a JavaScript expression to compute field's value, enclosed in quotes or backticks. (Can evaluate to a constant.) For details, see Authentication Fields.

4. Next, you can configure the following **Optional Settings**:
    - **Tags**: Optionally, add tags that you can use to filter and group Sources in Cribl Edge's UI. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
5. Optionally, you can adjust the TLS, Persistent Queue Settings, Processing and Advanced settings, or Connected Destinations outlined in the sections below.
6. Select **Save**, then **Commit & Deploy**.

Authentication Fields

By default, fields specified here will override fields of the same name in incoming events. However, you can customize this behavior to allow event fields to take precedence. For example, to allow the event's `index` field to take precedence, you could use this expression: `` `event.index || 'defaultIndex'` ``.

Fields add context to events based on the token used. This can be useful for tagging events with metadata such as source or environment, enhancing security by avoiding direct use of authorization values in Routes and Pipelines.

TLS Settings (Server Side)



On Cribl.Cloud, you can enable TLS using Cribl-provided certficates if desired.

Enabled: Defaults to toggled off. When toggled on:

Certificate: Name of the predefined certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

Passphrase: Passphrase to use to decrypt private key.

Certificate path: Server path containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

CA certificate path: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

Authenticate client (mutual auth): Require clients to present their certificates. Used to perform mutual authentication using SSL certs. Default is toggled off. When toggled on:

- **Validate client certificates**: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (for example, the system's CA). Default is toggled on.
- **Common name**: Regex that a peer certificate's subject attribute must match in order to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters.

(For example, to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.) If the subject attribute contains Subject Alternative Name (SAN) entries, the Source will check the regex against all of those but ignore the Common Name (CN) entry (if any). If the certificate has no SAN extension, the Source will check the regex against the single name in the CN.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In the **Persistent Queue Settings** tab, you can optionally specify persistent queue storage, using the following controls. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Source Persistent Queues (sPQ)

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the **Enable persistent queue** toggle. If enabled, PQ is automatically configured in `Always On` mode, with a maximum queue size of 1 GB disk space allocated per PQ-enabled Source, per Worker Process.

The 1 GB limit is on uncompressed inbound data, and the queue does not perform any compression. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable persistent queue: Default is toggled off. When toggled on:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to the Cribl Stream data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from the Cribl Stream data processing engine.

`Smart` mode only engages when necessary, such as when a downstream Destination becomes blocked *and* the **Buffer size limit** reaches its limit. When persistent queue is set to `Smart` mode, Cribl attempts to flush the queue when every new event arrives. The only time events stay in the buffer is when a downstream Destination becomes blocked.

Buffer size limit: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. This buffer is for all connections, not just per Worker Process. For that reason, this can dramatically expand memory usage. Connections share this limit, which may result in slightly lower throughput for higher numbers of connections. For higher numbers of connections, consider increasing the limit.

Commit frequency: The number of events to send downstream before committing that Stream has read them. Defaults to `42`.

File size limit: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, and so forth. If not specified, Cribl Stream applies the default `1 MB`.

Queue size limit: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, and so forth. Can be set as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** in Group or Fleet settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file closes. Defaults to `None`; `Gzip` is also available.



In Cribl Stream 4.1 and newer, the Source persistent queue default **Mode** is `Always On`, to best ensure events' delivery. For details on optimizing this selection, see Optimize Source Persistent Queues (sPQ).

You can optimize Workers' startup connections and CPU load at **Group/Fleet settings** > Worker Processes.

Processing Settings Event Breakers

Event Breaker rulesets: A list of event breaking rulesets that will be applied to the input data stream before the data is sent through the Routes. Defaults to `System Default Rule`.

Event Breaker buffer timeout: How long (in milliseconds) the Event Breaker will wait for new data to be sent to a specific channel, before flushing out the data stream, as-is, to the Routes. Minimum `10` ms, default `10000` (10 sec), maximum `43200000` (12 hours).

Fields

In this section, you can define new fields or modify existing ones using JavaScript expressions, similar to the Eval function.

- The **Field Name** can either be a new field (unique within the event) or an existing field name to modify its value.
- The **Value** is a JavaScript expression (enclosed in quotes or backticks) to compute the field's value (can be a constant). Select this field's advanced mode icon (far right) if you'd like to open a modal where you can work with sample data and iterate on results.

This flexibility means you can:

- Add new fields to enrich the event.
- Modify existing fields by overwriting their values.
- Compute logic or transformations using JavaScript expressions.

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline or Pack to process data from this input before the data is sent through the Routes.

Advanced Settings

Show originating IP: Toggle on when clients are connecting through a proxy that supports the `X-Forwarded-For` header to keep the client's original IP address on the event instead of the proxy's IP address. This setting affects how the Source handles the `__srcIpPort` field.

Capture request headers: Toggle on to add request headers to events, in the `__headers` field.

Health check endpoint: Toggle on to enable a health check endpoint specific to this Source, `http(s)://<host>:<port>/cribl_health`. A `200` HTTP response code is returned when the Source is healthy. Otherwise, two errors you could receive are:

- `ECONNRESET` where the Source failed to initialize due to not having listeners on the port.
- `503` or `Server is busy, max active connections reached` indicate there are too many connections per Worker Process.

Allowed URI paths: List of URI paths accepted by this input. Supports wildcards, for example, `/api/v*/hook`. Defaults to `*`, which allows all paths.

Allowed HTTP methods: List of HTTP methods accepted by this input. Supports wildcards, for example, `P*`, `GET`. Defaults to `*`, which allows all methods.

Active request limit (on-prem only): Maximum number of active requests allowed for this Source, per Worker Process. Defaults to `256`. Enter `0` for unlimited.

Raising this limit can increase throughput by allowing more concurrent data requests, but increases resource usage and load on both your Cribl Edge infrastructure and on downstream Destinations. Before raising this limit, ensure:

- Your Cribl Edge deployment has sufficient capacity to support higher request concurrency, including CPU, memory, or number of Worker Processes.
- Downstream Destinations are correctly sized and tuned to accept the higher data ingest rate, preventing backpressure. See Manage Backpressure for more information.

Improper sizing on either side can result in dropped events, delayed processing, or overall system instability.

Activity log sample rate: Determines how often request activity is logged at the `info` level. The default `100` value logs every 100th value; a `1` value would log every request; a `10` value would log every 10th request; etc.

Requests-per-socket limit: The maximum number of requests Cribl Edge should allow on one socket before instructing the client to close the connection. Defaults to `0` (unlimited). See Balancing Connection Reuse Against Request Distribution below.

Socket timeout (seconds): How long Cribl Edge should wait before assuming that an inactive socket has timed out. The default `0` value means wait forever.

Request timeout (seconds): How long to wait for an incoming request to complete before aborting it. The default `0` value means wait indefinitely.

Keep-alive timeout (seconds): After the last response is sent, Cribl Edge will wait this long for additional data before closing the socket connection. Defaults to `5` seconds; minimum is `1` second; maximum is `600` seconds (10 minutes).

The longer the **Keep-alive timeout**, the more Cribl Edge will reuse connections. The shorter the timeout, the closer Cribl Edge gets to creating a new connection for every request. When request frequency is high, you can use longer timeouts to reduce the number of connections created, which mitigates the associated cost.

IP allowlist regex: Grants access to requests originating from specific IP addresses that match a defined pattern. Unmatched requests are rejected with a 403 (Forbidden) status code. Defaults to `.*` (allow all).

IP denylist regex: Blocks requests originating from specific IP addresses that match a defined pattern, even if they would be allowed by default. Rejected requests receive a 403 (Forbidden) status code. Defaults to `^$` (allow all).

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Balancing Connection Reuse Against Request Distribution

Requests-per-socket limit allows you to limit the number of HTTP requests an upstream client can send on one network connection. Once the limit is reached, Cribl Edge uses HTTP headers to inform the client that it must establish a new connection to send any more requests. (Specifically, Cribl Edge sets the HTTP `Connection` header to `close`.) After that, if the client disregards what Cribl Edge has asked it to do and tries to send another HTTP request over the existing connection, Cribl Edge will respond with an HTTP status code of `503 Service Unavailable`.

Use this setting to strike a balance between connection reuse by the client, and distribution of requests among one or more Edge Node processes by Cribl Edge:

- When a client sends a sequence of requests on the same connection, that is called connection reuse. Because connection reuse benefits client performance by avoiding the overhead of creating new connections, clients have an incentive to **maximize** connection reuse.
- Meanwhile, a single process on that Edge Node will handle all the requests of a single network connection, for the lifetime of the connection. When receiving a large overall set of data, Cribl Edge performs better when the workload is distributed across multiple Edge Node processes. In that situation, it makes sense to **limit** connection reuse.

There is no one-size-fits-all solution, because of variation in the size of the payload a client sends with a request and in the number of requests a client wants to send in one sequence. Start by estimating how long connections will stay open. To do this, multiply the typical time that requests take to process (based on payload size) times the number of requests the client typically wants to send.

If the result is 60 seconds or longer, set **Requests-per-socket limit** to force the client to create a new connection sooner. This way, more data can be spread over more Edge Node processes within a given unit of time.

For example: Suppose a client tries to send thousands of requests over a very few connections that stay open for hours on end. By setting a relatively low **Requests-per-socket limit**, you can ensure that the same work is done over more, shorter-lived connections distributed between more Edge Node processes, yielding better performance from Cribl Edge.

A final point to consider is that one Cribl Edge Source can receive requests from more than one client, making it more complicated to determine an optimal value for **Requests-per-socket limit**.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

Internal Fields

Cribl Edge uses a set of internal fields to assist in handling of data. These "meta" fields are **not** part of an event, but they are accessible, and functions can use them to make processing decisions.

Fields for this Source:

- `__channel`
- `__headers` - Automatically includes any headers sent with request.
- `__inputId`
- `__srcIpPort` - See details below.

Overriding `__srcIpPort` with Client IP/Port

The `__srcIpPort` field's value contains the IP address and (optionally) port of the HTTP client sending data to this Source.

When any proxies (including load balancers) lie between the HTTP client and the Source, the last proxy adds an `X-Forwarded-For` header whose value is the IP/port of the original client. With multiple proxies, this header's value will be an array, whose first item is the original client IP/port.

If `X-Forwarded-For` is present, and **Advanced Settings** > **Show originating IP** is toggled off, the original client IP/port in this header will override the value of `__srcIpPort`.

If **Show originating IP** is toggled on, the `X-Forwarded-For` header's contents will **not** override the `__srcIpPort` value. (Here, the upstream proxy can convey the client IP/port without using this header.)

Examples

The examples in this section demonstrate sending HTTP data into a Raw HTTP Source. The token is set to `myToken42` and included in the `Authorization` header of requests.

Cribl Endpoint - Single Event

Replace `<myCriblHost>` with your **Public Ingress address**.

Cribl Endpoint, Single Event Example

```
curl -k "http://<myCriblHost>:10050/postPath" \
-H "Authorization: myToken42" \
-H "Content-Type: application/json" \
-d '{"_raw":"this is a sample event", "host":"myHost", "source":"mySource", "fieldA":
```

Copy

Cribl Endpoint - Multiple Events

Replace `<myCriblHost>` with your Public Ingress address.

Cribl Endpoint, Multiple Events Example

```
curl -k "http://<myCriblHost>:10050/postPath" \
-H "Authorization: myToken42" \
-H "Content-Type: application/json" \
-d $'{"_raw":"this is a sample event", "host":"myHost", "source":"mySource", "fieldA'
{"_raw":"this is another sample event", "host":"myOtherHost", "source":"myOtherSource
```

Copy

Cribl.Cloud - Single Event

Replace `<Your-Org-ID>` with your Cribl.Cloud Organization ID. Cribl.Cloud makes the `20000` - `20010` port range available for configuring Sources that don't have a default port, like the Raw HTTP Source.

Cribl.Cloud, Single Event Example

```
curl -k "http://default.main.<Your-Org-ID>.cribl.cloud:20001" \
-H "Authorization: myToken42" \
-d '{"_raw":"this is a sample event"}'
```

Copy

For Cribl.Cloud Enterprise plans, replace `default` in the request URL with the name of the Fleet that you want to send to.

Troubleshooting

The Source's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they are ingested. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the ingestion process, including any errors or warnings that may have occurred.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify ingestion issues. Analyze the graphs showing events and bytes in/out over time.

Common Issue

Dropping request because token invalid",“authToken”: “Bas...Njc=”

The specified token is invalid. Note that the above message is logged only at the debug level.

12.2.13. Metrics Source

Cribl Edge supports receiving metrics in these wire formats/protocols: StatsD, StatsD Extended, and Graphite. Automatic protocol detection happens on the first line received over a TCP connection or a UDP packet. Lines not matching the detected protocol are dropped.

Type: **Push** | TLS Support: **No** | Event Breaker Support: **No**

Cribl Edge Workers support System Metrics only when running on Linux, not on Windows.

Configure Cribl Edge to Receive Metrics

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Source** and select the Source you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). Select the Source you want. Next, select **Add Source**.
2. In the **New Source** modal, configure the following under **General Settings**:
    - **Input ID**: Enter a unique name to identify this Source definition. If you clone this Source, Cribl Edge will add `-CLONE` to the original **Input ID**.
    - **Description**: Optionally, enter a description.
    - **Address**: Enter the hostname/IP to listen to. Defaults to `0.0.0.0`.
    - **UDP port**: Enter the UDP port number to listen on. Not required if listening on TCP.For detail, see UDP Port Considerations.
    - **TCP port**: Enter the TCP port number to listen on. Not required if listening on UDP.
3. Next, you can configure the following **Optional Settings**:
    - **Tags**: Optionally, add tags that you can use to filter and group Sources in Cribl Edge's UI. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
4. Optionally, you can adjust the TLS, Persistent Queue Settings, Processing and Advanced settings, or Connected Destinations outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

UDP Port Considerations

Sending large numbers of UDP events per second can cause Cribl.Cloud to drop some of the data. This results from restrictions of the UDP protocol. To minimize the risk of data loss, deploy a customer-managed (hybrid) Stream Worker Group with Worker Nodes as close as possible to the UDP senders. Cribl also recommends tuning the OS UDP buffer size.

TLS Settings (TCP Only)

Enabled: Defaults to toggled off. When toggled on:

Certificate: Name of the predefined certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

Passphrase: Passphrase to use to decrypt private key.

Certificate path: Server path containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

CA certificate path: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

Authenticate client (mutual auth): Require clients to present their certificates. Used to perform mutual authentication using SSL certs. Default is toggled off. When toggled on:

- **Validate client certificates**: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (for example, the system's CA). Default is toggled on.
- **Common name**: Regex that a peer certificate's subject attribute must match in order to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. (For example, to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.) If the subject attribute contains Subject Alternative Name (SAN) entries, the Source will check the regex against all of those but ignore the Common Name (CN) entry (if any). If the certificate has no SAN extension, the Source will check the regex against the single name in the CN.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In the **Persistent Queue Settings** tab, you can optionally specify persistent queue storage, using the following controls. Persistent queue buffers and preserves incoming events when a downstream Destination has an

outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Source Persistent Queues (sPQ)

> On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the **Enable persistent queue** toggle. If enabled, PQ is automatically configured in `Always On` mode, with a maximum queue size of 1 GB disk space allocated per PQ-enabled Source, per Worker Process.
>
> The 1 GB limit is on uncompressed inbound data, and the queue does not perform any compression. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable persistent queue: Default is toggled off. When toggled on:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to the Cribl Stream data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from the Cribl Stream data processing engine.

> `Smart` mode only engages when necessary, such as when a downstream Destination becomes blocked *and* the **Buffer size limit** reaches its limit. When persistent queue is set to `Smart` mode, Cribl attempts to flush the queue when every new event arrives. The only time events stay in the buffer is when a downstream Destination becomes blocked.

Buffer size limit: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. This buffer is for all connections, not just per Worker Process. For that reason, this can dramatically expand memory usage. Connections share this limit, which may result in slightly lower throughput for higher numbers of connections. For higher numbers of connections, consider increasing the limit.

Commit frequency: The number of events to send downstream before committing that Stream has read them. Defaults to `42`.

File size limit: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, and so forth. If not specified, Cribl Stream applies the default `1 MB`.

Queue size limit: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data. Required, and defaults to `5 GB`. Accepts positive numbers with units of `KB`, `MB`, `GB`, and so forth. Can be set as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** in Group or Fleet settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file closes. Defaults to `None`; `Gzip` is also available.



In Cribl Stream 4.1 and newer, the Source persistent queue default **Mode** is `Always On`, to best ensure events' delivery. For details on optimizing this selection, see Optimize Source Persistent Queues (sPQ).

You can optimize Workers' startup connections and CPU load at **Group/Fleet settings** > Worker Processes.

Processing Settings Fields

In this section, you can define new fields or modify existing ones using JavaScript expressions, similar to the Eval function.

- The **Field Name** can either be a new field (unique within the event) or an existing field name to modify its value.
- The **Value** is a JavaScript expression (enclosed in quotes or backticks) to compute the field's value (can be a constant). Select this field's advanced mode icon (far right) if you'd like to open a modal where you can work with sample data and iterate on results.

This flexibility means you can:

- Add new fields to enrich the event.
- Modify existing fields by overwriting their values.
- Compute logic or transformations using JavaScript expressions.

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline or Pack to process data from this input before the data is sent through the Routes.

Advanced Settings

Enable Proxy Protocol: Toggle on if the connection is proxied by a device that supports Proxy Protocol v1 or v2.

IP allowlist regex: Regex matching IP addresses that are allowed to send data. Defaults to `.*` (i.e., all IPs.)

Buffer size limit (events): Maximum number of events to buffer when downstream is blocking. Defaults to `1000`.

UDP socket buffer size (bytes): Optionally, set the SO_RCVBUF socket option for the UDP socket. This value tells the operating system how many bytes can be buffered in the kernel before events are dropped. Leave blank to use the OS default. Min: `256`. Max: `4294967295`.

It may also be necessary to increase the size of the buffer available to the `SO_RCVBUF` socket option. Consult the documentation for your operating system for a specific procedure.



Setting this value will affect OS memory utilization.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

Internal Fields

Cribl Edge uses a set of internal fields to assist in handling of data. These "meta" fields are **not** part of an event, but they are accessible, and Functions can use them to make processing decisions.

Fields for this Source:

- `__srcIpPort`
- `__metricsInType`

Metric Event Schema and Destination Support

Metric data is read into the following event schema:

```
_metric - the metric name
_metric_type - the type of the metric (gauge, counter, timer)
_value - the value of the metric
_time - metric_time or Date.now()/1000
dim1 - value of dimension1
dim2 - value of dimension2
....
```

Copy

Cribl Edge places sufficient information into a field called `__criblMetric` to enable these events to be properly serialized out to any metric outputs (independent of the input type).

The following Destinations natively support the `__criblMetric` field:

- Splunk
- Splunk HEC
- InfluxDB
- Statsd
- Statsd Extended
- Graphite

Data Format/Protocol Examples StatsD

Format: `MetricName:value|type`

StatsD Example

```
metric1:100|g
metric2:200|ms
metric.dot.3:300.16|c
```

Copy

See the StatsD repo.

StatsD Extended

Format: `MetricName:value|type|#dim=value,dim2=value`

StatsD Extended Example

```
metric1:100|g|#dim1:val1,dim2:val2,dim3:val3
metric2:200|ms|#dim1:val1,dim2:val2,dim3:val3
metric.dot.3:300.16|c|#dim1:val1,dim2:val2,dim3:val3
```

Copy

Graphite

Format: `MetricName[;dim1=val1[;dim2=val2]] value time`

Graphite Example with Dimensions

```
metric1;dim1=val1;dim2=val2 100 9999
metric2;dim1=val1;dim2=val2 200 9999
metric.dot.3;dim1=val1;dim2=val2 300.16 9999.16
```

Copy

Graphite Example without Dimensions

```
metric1 100 9999
metric2 200 9999
metric.dot.3 300.16 9999.16
```

Copy

See the Graphite (also known as Carbon) plaintext protocol.

Troubleshooting

The Source's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they are ingested. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the ingestion process, including any errors or warnings that may have occurred.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify ingestion issues. Analyze the graphs showing events and bytes in/out over time.

12.2.14. Model Driven Telemetry Source

Cribl Edge supports receiving network device metrics and events via Model Driven Telemetry (MDT).



Type: **Push** | TLS Support: **Yes** | Event Breaker Support: **No**

For observability of network devices, MDT compares favorably with the older SNMP-based approach because:

- MDT's messaging granularity is much finer than SNMP's.
- MDT's push-based model reduces load on network devices, while SNMP's pull-based model queues requests on devices.
- MDT's extensive ecosystem of data modules enables devices to collect many more kinds of data than they could with SNMP tools.

Prerequisites

When you configure MDT on a network device you choose among a variety of options and techniques. For this Source (in Cribl Edge version 4.5), you must configure your upstream sending device to use the following:

- MDT in dial-out mode.
- gRPC for the data transport mechanism.
- YANG models for data representation.
- Self-describing, also known as Key/Value Protocol Buffers (protobufs) for data encoding.

Configure Cribl Edge to Receive Model Driven Telemetry Data

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect**. Select **Add Source** and select the Source you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Sources**. Select the Source you want. Next, select **Add Source**.
2. In the **New Source** modal, configure the following under **General Settings**:

- **Input ID**: Enter a unique name to identify this Source definition. If you clone this Source, Cribl Edge will add `-CLONE` to the original **Input ID**.
- **Description**: Optionally, enter a description.
- **Address**: Enter the hostname/IP to listen to. Defaults to `0.0.0.0`.
- **Port**: Enter the port number to listen on. Defaults to port `57000`.

3. Under **Authentication**, enter the **Auth tokens**:
   - **Auth tokens**: Shared secrets to be provided by any client. Click **Generate** to create a new secret. If empty, permits open access.
   - **Description**: Optionally, enter a description.
   - **Fields**: Fields to add to events referencing this token. Each field is a **Name/Value** pair, where **Value** is a JavaScript expression to compute field's value, enclosed in quotes or backticks. (Can evaluate to a constant.) For details, see Authentication Fields.
4. Next, you can configure the following **Optional Settings**:
   - **Tags**: Optionally, add tags that you can use to filter and group Sources in Cribl Edge's UI. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
5. Optionally, you can adjust the TLS, Persistent Queue Settings, Processing and Advanced settings, or Connected Destinations outlined in the sections below.
6. Select **Save**, then **Commit & Deploy**.

TLS Settings (Server Side)

Enabled: Defaults to toggled off. When toggled on:

Certificate: Name of the predefined certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

Passphrase: Passphrase to use to decrypt private key.

Certificate path: Server path containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

CA certificate path: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

Authenticate client (mutual auth): Require clients to present their certificates. Used to perform mutual authentication using SSL certs. Default is toggled off. When toggled on:

- **Validate client certificates**: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (such as the system's CA). Default is toggled on.

- **Common name**: Regex that a peer certificate's subject attribute must match in order to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. (For example, to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.) If the subject attribute contains Subject Alternative Name (SAN) entries, the Source will check the regex against all of those but ignore the Common Name (CN) entry (if any). If the certificate has no SAN extension, the Source will check the regex against the single name in the CN.

Persistent Queue Settings

In the **Persistent Queue Settings** tab, you can optionally specify persistent queue storage, using the following controls. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Source Persistent Queues (sPQ)

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the **Enable persistent queue** toggle. If enabled, PQ is automatically configured in `Always On` mode, with a maximum queue size of 1 GB disk space allocated per PQ-enabled Source, per Worker Process.

The 1 GB limit is on uncompressed inbound data, and the queue does not perform any compression. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable persistent queue: Default is toggled off. When toggled on:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to the Cribl Stream data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from the Cribl Stream data processing engine.

`Smart` mode only engages when necessary, such as when a downstream Destination becomes blocked *and* the **Buffer size limit** reaches its limit. When persistent queue is set to `Smart` mode, Cribl attempts to flush the queue when every new event arrives. The only time events stay in the buffer is when a downstream Destination becomes blocked.

Buffer size limit: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. This buffer is for all connections, not just per Worker Process. For that reason, this can dramatically expand memory usage. Connections share this limit, which may result in slightly lower throughput for higher numbers of connections. For higher numbers of connections, consider increasing the limit.

Commit frequency: The number of events to send downstream before committing that Stream has read them. Defaults to `42`.

File size limit: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, and so forth. If not specified, Cribl Stream applies the default `1 MB`.

Queue size limit: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, and so forth. Can be set as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** in Group or Fleet settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file closes. Defaults to `None`; `Gzip` is also available.



In Cribl Stream 4.1 and newer, the Source persistent queue default **Mode** is `Always On`, to best ensure events' delivery. For details on optimizing this selection, see Optimize Source Persistent Queues (sPQ).

You can optimize Workers' startup connections and CPU load at **Group/Fleet settings** > Worker Processes.

Processing Settings Fields

In this section, you can define new fields or modify existing ones using JavaScript expressions, similar to the Eval function.

- The **Field Name** can either be a new field (unique within the event) or an existing field name to modify its value.
- The **Value** is a JavaScript expression (enclosed in quotes or backticks) to compute the field's value (can be a constant). Select this field's advanced mode icon (far right) if you'd like to open a modal where you can work with sample data and iterate on results.

This flexibility means you can:

- Add new fields to enrich the event.

- Modify existing fields by overwriting their values.
- Compute logic or transformations using JavaScript expressions.

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline or Pack to process data from this input before the data is sent through the Routes.

Advanced Settings

Active connection limit: Maximum number of active connections allowed per Worker Process. Defaults to `1000`. Set a lower value if connection storms are causing the Source to hang. Set `0` for unlimited connections.

Shutdown timeout: Time to allow the server to shut down gracefully before forcing shutdown. Defaults to `5000` milliseconds (5 seconds).

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of data from this Source via the Routing table.

Select **QuickConnect** to send data from this Source to one or more Destinations via independent, direct connections.

Internal Fields

Cribl Edge uses a set of internal fields to assist in handling of data. These "meta" fields are **not** part of an event, but they are accessible, and Functions can use them to make processing decisions.

Fields for this Source:

- `__srcIpPort`

Troubleshooting

The Source's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they are ingested. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the ingestion process, including any errors or warnings that may have occurred.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify ingestion issues. Analyze the graphs showing events and bytes in/out over time.

12.2.15. OpenTelemetry (OTel) Source

Cribl Edge supports receiving trace, metric, and log events from OTLP-compliant senders.



Type: **Push** | TLS Support: **YES** | Event Breaker Support: **No**

Looking for the OpenTelemetry Source in **Cribl Search**? See Ingest OpenTelemetry Data into Cribl Search.

Supported and Unsupported Input Data

The OpenTelemetry (OTel) Source supports receiving:

- Compressed inbound data (with DEFLATE or gzip compression), as well as uncompressed data.
- Telemetry data over either of the transports that the OpenTelemetry Protocol (OTLP) describes: gRPC or HTTP. OTLP defines Protocol buffer (Protobuf) schemas for its payloads (requests and responses). With the HTTP transport, this Source supports Binary Protobuf payload encoding, but does not support JSON Protobuf.

The OpenTelemetry Project's Signals documentation provides these hierarchical definitions of Cribl Edge's supported trace and metric event types:

- A **trace** tracks the progression of a single request.
    - Each trace is a tree of **spans**.
    - A span object represents the work being done by the individual services, or components, involved in a request as that request flows through a system.
- A **metric** provides aggreggated statistical information.
    - A metric contains individual measurements called **data points**.
- A **log** provides a recording of an event.
    - See the OTLP Logs Function topic for an explanation of OTLP logs structure.

Breaking Changes Between OTLP 0.19.0 and 1.3.1

With version 0.19.0, the OTLP Protobuf spec introduced breaking changes in its field names. This means that if you created Routes or Pipelines that work with OTLP version 0.10.0, they may break when you use them with OTLP 1.3.1.

To use Routes or Pipelines created before Cribl Edge 4.7 with OTLP 1.3.1, you must check their filters and logic for any use of obsolete field names, and modify them accordingly,

Configure an OTel Source

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Source** and select the Source you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). Select the Source you want. Next, select **Add Source**.
2. In the **New Source** modal, configure the following under **General Settings**:
    - **Input ID**: Unique ID for this Source. For example, `OTel042`.
    - **Description**: Optionally, enter a description.
    - **OTLP version**: The drop-down offers `0.10.0` and `1.3.1` (default).
    - **Protocol**: Use the drop-down to choose the protocol matching the data you will ingest: `gRPC` (default), or `HTTP`.
    - **Address**: Enter the hostname/IP to listen on. Defaults to `0.0.0.0` (all addresses, IPv4 format).
    - **Port**: OTLP uses port `4317` for gRPC and `4318` for HTTP. This Source defaults to `4317` when **Protocol** is gRPC (default) and to `4318` when **Protocol** is HTTP.
    - The **Extract spans**, **Extract metrics**, and **Extract logs** settings are specific to the OpenTelemetry Source. By default, these options are toggled off, allowing Cribl Edge to act as a pass-through that generates a single event for each incoming OTel payload. This is useful when you want to forward complete OTel events to downstream systems, such as persistent storage, without breaking them apart. You can enable these settings to extract and process individual records from within OTel events:
        - **Extract spans**: Generates an individual event for each span contained in an OTLP traces export payload. A single OTLP traces payload (batch) typically includes many spans.
        - **Extract metrics**: Generates an individual event for each data point in a metric event. OTel metrics often contain multiple data points per event.
        - **Extract logs**: Available only when **OTLP version** is set to `1.3.1`. Generates an individual event for each log record. Cribl recommends enabling this option to simplify log transformation and manipulation.

    - **Tags**: Optionally, add tags to help filter and group Sources within Cribl Edge's UI. Tags are not included in the event data. Separate tag names using a tab or hard return.

3. Optionally, you can adjust the Authentication, TLS, Persistent Queue Settings, Processing, and Advanced settings, or Connected Destinations outlined in the following sections.
4. Select **Save**, then **Commit & Deploy**.

Authentication

Select one of the following options for authentication:

- **None**: Don't use authentication.
- **Auth token**: Enter the bearer token that must be included in the authorization header.
- **Auth token (text secret)**: Provide an HTTP token referenced by a secret. Select a stored text secret in the resulting drop-down, or click **Create** to configure a new secret.
- **Basic**: Displays **Username** and **Password** fields for you to enter HTTP Basic authentication credentials.
- **Basic (credentials secret)**: Provide username and password credentials referenced by a secret. Select a stored text secret in the resulting **Credentials secret** drop-down, or click **Create** to configure a new secret.

TLS Settings (Server Side)



In OTel terminology, your Cribl Edge OTel Source will receive OTel data from a **Collector** running on a local **agent**. In Cribl Edge's terminology, the Collector is the **client** and the OTel Source is the **server**. This is why this Source's UI identifies the Source's TLS Settings as "server-side."

For more about this client-server relationship, see the TLS Configuration Example below.

On Cribl.Cloud, you can enable TLS using Cribl-provided certficates if desired.

Enabled: Defaults to toggled off. When toggled on:

Certificate: Name of the predefined certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

Passphrase: Passphrase to use to decrypt private key.

Certificate path: Server path containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

CA certificate path: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

Authenticate client (mutual auth): Require clients to present their certificates. Used to perform mutual authentication using SSL certs (mTLS). Default is toggled off. When toggled on:

- **Validate client certificates**: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (for example, the system's CA). Default is toggled on.
- **Common name**: Regex that a peer certificate's subject attribute must match in order to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. (For example, to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.) If the subject attribute contains Subject Alternative Name (SAN) entries, the Source will check the regex against all of those but ignore the Common Name (CN) entry (if any). If the certificate has no SAN extension, the Source will check the regex against the single name in the CN.

Minimum TLS version: Defines the oldest TLS version allowed for incoming connections. Connections using an earlier version will be rejected. Use this setting to enforce a baseline for secure communication.

Maximum TLS version: Defines the newest TLS version that can be negotiated during a connection. Use this setting to restrict connections to tested and supported TLS versions.

Persistent Queue Settings

In the **Persistent Queue Settings** tab, you can optionally specify persistent queue storage, using the following controls. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Source Persistent Queues (sPQ)

> On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the **Enable persistent queue** toggle. If enabled, PQ is automatically configured in `Always On` mode, with a maximum queue size of 1 GB disk space allocated per PQ-enabled Source, per Worker Process.
>
> The 1 GB limit is on uncompressed inbound data, and the queue does not perform any compression. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable persistent queue: Default is toggled off. When toggled on:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to the Cribl Stream data processing engine.

- `Smart`: This option will engage PQ only when the Source detects backpressure from the Cribl Stream data processing engine.



`Smart` mode only engages when necessary, such as when a downstream Destination becomes blocked *and* the **Buffer size limit** reaches its limit. When persistent queue is set to `Smart` mode, Cribl attempts to flush the queue when every new event arrives. The only time events stay in the buffer is when a downstream Destination becomes blocked.

Buffer size limit: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. This buffer is for all connections, not just per Worker Process. For that reason, this can dramatically expand memory usage. Connections share this limit, which may result in slightly lower throughput for higher numbers of connections. For higher numbers of connections, consider increasing the limit.

Commit frequency: The number of events to send downstream before committing that Stream has read them. Defaults to `42`.

File size limit: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, and so forth. If not specified, Cribl Stream applies the default `1 MB`.

Queue size limit: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, and so forth. Can be set as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** in Group or Fleet settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file closes. Defaults to `None`; `Gzip` is also available.



In Cribl Stream 4.1 and newer, the Source persistent queue default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Optimize Source Persistent Queues (sPQ).

You can optimize Workers' startup connections and CPU load at **Group/Fleet settings** > Worker Processes.

Processing Settings Fields

In this section, you can define new fields or modify existing ones using JavaScript expressions, similar to the Eval function.

- The **Field Name** can either be a new field (unique within the event) or an existing field name to modify its value.
- The **Value** is a JavaScript expression (enclosed in quotes or backticks) to compute the field's value (can be a constant). Select this field's advanced mode icon (far right) if you'd like to open a modal where you can work with sample data and iterate on results.

This flexibility means you can:

- Add new fields to enrich the event.
- Modify existing fields by overwriting their values.
- Compute logic or transformations using JavaScript expressions.

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline or Pack to process data from this input before the data is sent through the Routes.

Advanced Settings

When **General Settings** > **Protocol** is set to `gRPC`, the UI displays one additional setting:

- **Active connection limit**: Maximum number of active connections allowed per Worker Process. Defaults to `1000`. Set a lower value if connection storms are causing the Source to hang. Set `0` for unlimited connections.
- **Environment**: Optionally, specify a single Git branch on which to enable this configuration. If this field is empty, the config will be enabled everywhere.

When **General Settings** > **Protocol** is set to `HTTP`, the UI displays these additional settings:

- **Health check endpoint**: Toggle on to enable a health check endpoint specific to this Source, `http(s)://<host>:<port>/cribl_health`. A `200` HTTP response code is returned when the Source is healthy. Otherwise, two errors you could receive are:
  - `ECONNRESET` where the Source failed to initialize due to not having listeners on the port.
  - `503` or `Server is busy, max active connections reached` indicate there are too many connections per Worker Process.
- **Active request limit (on-prem only)**: Maximum number of active requests allowed for this Source, per Worker Process. Defaults to `256`. Enter `0` for unlimited.



Raising this limit can increase throughput by allowing more concurrent data requests, but increases resource usage and load on both your Cribl Edge infrastructure and on downstream Destinations.

Before raising this limit, ensure:

- Your Cribl Edge deployment has sufficient capacity to support higher request concurrency, including CPU, memory, or number of Worker Processes.
- Downstream Destinations are correctly sized and tuned to accept the higher data ingest rate, preventing backpressure. See Manage Backpressure for more information.

Improper sizing on either side can result in dropped events, delayed processing, or overall system instability.

- **Requests-per-socket limit**: The maximum number of requests Cribl Edge should allow on one socket before instructing the client to close the connection. Defaults to `0` (unlimited). See Balancing Connection Reuse Against Request Distribution below.
- **Socket timeout (seconds)**: How long Cribl Edge should wait before assuming that an inactive socket has timed out. The default `0` value means wait forever.
- **Request timeout (seconds)**: How long to wait for an incoming request to complete before aborting it. The default `0` value means wait indefinitely.
- **Keep-alive timeout (seconds)**: After the last response is sent, Cribl Edge will wait this long for additional data before closing the socket connection. Defaults to `5` seconds; minimum is `1` second; maximum is `600` seconds (10 minutes).
- **IP allowlist regex**: Grants access to requests originating from specific IP addresses that match a defined pattern. Unmatched requests are rejected with a 403 (Forbidden) status code. Defaults to `.*` (allow all).
- **IP denylist regex**: Blocks requests originating from specific IP addresses that match a defined pattern, even if they would be allowed by default. Rejected requests receive a 403 (Forbidden) status code. Defaults to `^$` (allow all).
- **Environment**: Optionally, specify a single Git branch on which to enable this configuration. If this field is empty, the config will be enabled everywhere.

Balancing Connection Reuse Against Request Distribution

Requests-per-socket limit allows you to limit the number of HTTP requests an upstream client can send on one network connection. Once the limit is reached, Cribl Edge uses HTTP headers to inform the client that it must establish a new connection to send any more requests. (Specifically, Cribl Edge sets the HTTP `Connection` header to `close`.) After that, if the client disregards what Cribl Edge has asked it to do and tries to send another HTTP request over the existing connection, Cribl Edge will respond with an HTTP status code of `503 Service Unavailable`.

Use this setting to strike a balance between connection reuse by the client, and distribution of requests among one or more Edge Node processes by Cribl Edge:

- When a client sends a sequence of requests on the same connection, that is called connection reuse. Because connection reuse benefits client performance by avoiding the overhead of creating new connections, clients have an incentive to **maximize** connection reuse.
- Meanwhile, a single process on that Edge Node will handle all the requests of a single network connection, for the lifetime of the connection. When receiving a large overall set of data, Cribl Edge performs better when the workload is distributed across multiple Edge Node processes. In that situation, it makes sense to **limit** connection reuse.

There is no one-size-fits-all solution, because of variation in the size of the payload a client sends with a request and in the number of requests a client wants to send in one sequence. Start by estimating how long connections will stay open. To do this, multiply the typical time that requests take to process (based on payload size) times the number of requests the client typically wants to send.

If the result is 60 seconds or longer, set **Requests-per-socket limit** to force the client to create a new connection sooner. This way, more data can be spread over more Edge Node processes within a given unit of time.

For example: Suppose a client tries to send thousands of requests over a very few connections that stay open for hours on end. By setting a relatively low **Requests-per-socket limit**, you can ensure that the same work is done over more, shorter-lived connections distributed between more Edge Node processes, yielding better performance from Cribl Edge.

A final point to consider is that one Cribl Edge Source can receive requests from more than one client, making it more complicated to determine an optimal value for **Requests-per-socket limit**.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

Internal Fields

OTel internal fields are metadata fields added to events processed by the OTel Source. These fields provide essential context about the event's origin and processing details and can be referenced in filters, Routes, and Pipelines to apply specific logic based on the event's metadata. These fields are not included in the final output unless explicitly configured at the Pipeline or Destination stage.

Fields for this Source:

- `__otlp.version` - The OTLP version used to parse the event. For example, `0.10.0` or `1.3.1`. This helps ensure compatibility with upstream systems and configurations.

- `__otlp.type` - The type of telemetry data. For example, `logs`, `metrics`, or `traces`. This is crucial for distinguishing between different signal types in Pipelines.
- `__otlp.extracted` - A boolean flag that denotes whether individual records were extracted from a batch event. If `true`, the event represents an extracted record. This behavior is directly governed by the **Extract spans**, **Extract metrics**, and **Extract logs** general settings of the OTel Source.

TLS Configuration Example

Here's a simple example for using TLS to secure the communication between an OpenTelemetry client and your Cribl Edge OTel Source.

1. Choose or generate a certificate and key. If you need to generate a certificate/key pair, you can adapt the following OpenSSL command:

   ```
   openssl req -nodes -new -x509 -newkey rsa:2048 -keyout myKey.pem -out myCert.pem
   ```

   Copy

   This example command will generate both a self-signed cert named `myCert.pem` (certified for 420 days), and an unencrypted, 2048-bit RSA private key named `myKey.pem`.

2. Configure the **TLS Settings (Server Side)**. Toggle **Enabled** on, then:
   - Enter the appropriate values in the **Certificate name**, **Private key path**, and **Certificate path** fields. A **Create** link is available if you need a new certificate, and **Certificate name** also works as a drop-down to allow you to choose from any existing certificates.
   - Leave the **CA certificate path** field empty.
   - Leave **Authenticate client (mutual auth)** toggled off.
3. Configure the OTel client. See the OTel Collector TLS Configuration Settings README for an explanation of the relevant settings. The config file might be named `otel-config.yaml`, `otel-local-config.yaml`, or just `config.yaml`, depending on your environment. This YAML file will have an `exporters` section, which you must edit to include an `otlp` sub-section, as follows:
   - Add an `endpoint` whose value is the IP address of either (a) the Cribl Edge Edge Node on which your OTel Source is running, or (b) the IP address of the load balancer for the relevant Fleet. In the example snippet below, this is the `<Cribl_IP_address>`. Specify the port on which Cribl Edge's OTel Source is listening; port `4317` is the OTLP default for gRPC (`4318` for HTTP).
   - Set `tls` > `insecure` to `false`. This matches your setting **TLS Settings (Server Side)** > **Enabled** to toggled on for the Cribl Edge OTel Source.
   - Set `tls` > `insecure_skip_verify` to `true`. This matches your setting **TLS Settings (Server Side)** > **Authenticate client (mutual auth)** to toggled off for the Cribl Edge OTel Source. Setting `insecure_skip_verify` to `true` is also required if you're using a self-signed certificate.

Here's how the section you edited should look:

```
exporters:
  otlp:
    endpoint: "https://<Cribl_IP_address>:4317"
    tls:
      insecure: false
      insecure_skip_verify: true
```

Copy

Troubleshooting

The Source's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they are ingested. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the ingestion process, including any errors or warnings that may have occurred.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify ingestion issues. Analyze the graphs showing events and bytes in/out over time.

Common Issue

When your upstream sender is the OpenTelemetry Collector and its logs show the error `Exporting failed. Will retry the request after interval`, try increasing the duration specified in **Advanced Settings** > **Keep-alive timeout (seconds)**. Choose a duration that aligns with your OpenTelemetry Collector configuration, such that the OpenTelemetry Collector has enough time to send its events before the keep-alive timeout elapses.

12.2.16. SNMP Trap Source

Cribl Edge supports receiving data from SNMP Traps.

Type: **Push** | TLS Support: **NO** | Event Breaker Support: **No**

Configure Cribl Edge to Receive SNMP Traps

Cribl Edge, except in Cribl.Cloud, ships with an SNMP Trap Source preconfigured to listen on port `9162`. You can clone or directly modify this Source to further configure it, and then enable it.

In Cribl Edge, configure SNMP Traps:

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Source** and select the Source you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). Select the Source you want. Next, select **Add Source**.
2. In the **New Source** modal, configure the following under **General Settings**:
    - **Input ID**: Enter a unique name to identify this Source definition. The default Source is prefilled with the value `in_snmp_trap`. which can't be changed via the UI. If you clone this Source, Cribl Edge will add `-CLONE` to the original **Input ID**.
    - **Description**: Optionally, enter a description.
    - **Address**: Address to bind on. Defaults to `0.0.0.0` (all addresses).
    - **UDP Port**: Port on which to receive SNMP traps. Defaults to `162`.

        Sending large numbers of UDP events per second can cause Cribl.Cloud to drop some of the data. This results from restrictions of the UDP protocol.

        To minimize the risk of data loss, deploy a customer-managed (hybrid) Stream Worker Group with Worker Nodes as close as possible to the UDP senders. Cribl also recommends tuning the OS UDP buffer size.

3. Next, you can configure the following **Optional Settings**:
    - **Tags**: Optionally, add tags that you can use to filter and group Sources in Cribl Edge's UI. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
4. Optionally, you can adjust the Authentication, Persistent Queue Settings, Processing and Advanced settings, or Connected Destinations outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

Authentication Settings

SNMPv3 authentication provides secure access to devices through optional user authentication and decryption of incoming data packets. Cribl Edge lets you choose any of the following levels of security:

- User name checking without authentication.
- User authentication without privacy.
- User authentication with privacy.

SNMPv3 authentication: Toggle on to enable SNMPv3 authentication and reveal authentication parameters. Default is toggled off.

Allow unmatched traps: Toggle on to pass through traps that don't match any of the configured users. Cribl Edge will not attempt to authenticate or decrypt these traps. When toggled off (default), Cribl Edge drops traps without a correctly configured user name.

v3 Name: Enter the SNMPv3 user name (required). Multiple users are supported.

You must configure at least one user to enable SNMPv3 authentication. If desired, you can enter separate user credentials for each SNMPv3 user. The authentication protocol, authentication key, privacy protocol, and privacy key can be unique for each user.

For authentication to succeed, the user name and authentication protocol and key must match in the Source and the sending device configuration.

For decryption to succeed, the user name, authentication protocol and key, and the privacy protocol and key must match in the Source and the sending device configuration.

Authentication protocol: Select the authentication protocol required for your use case. If you select `None`, Cribl Edge will only check the user name without authenticating the user or decrypting the data. Otherwise, Cribl Edge uses the authentication protocol you specify and the key you provide to authenticate the user.

Options include:

- `None`
- `MD5`

- SHA1
- SHA224
- SHA256
- SHA384
- SHA512

v3 authentication key: Enter the authentication key.

Choosing an authentication protocol also allows you to select an optional privacy protocol to decrypt incoming packets.

Cribl Edge logs authentication failures at the `debug` level.

Privacy protocol: Select the privacy protocol required for your use case. If you select `None`, Cribl Edge authenticates the user without decrypting the incoming data. Otherwise, Cribl Edge uses the privacy protocol you specify and the key you provide to decrypt the data.

Options include:

- `None`
- `AES128`
- `AES256b (Blumenthal)`
- `AES2556r (Reeder)`

For every successfully decrypted trap, Cribl Edge adds `__didDecrypt: true` to the Event. Cribl Edge logs decryption failures at the `debug` level.

v3 privacy key: Enter the privacy key.

Add user: Click to add a new set of user credentials for SNMPv3 authentication.

Persistent Queue Settings

In the **Persistent Queue Settings** tab, you can optionally specify persistent queue storage, using the following controls. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Source Persistent Queues (sPQ)

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the **Enable persistent queue** toggle. If enabled, PQ is automatically configured in `Always On` mode, with a maximum queue size of 1 GB disk space allocated per PQ-enabled Source, per Worker Process.

The 1 GB limit is on uncompressed inbound data, and the queue does not perform any compression. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable persistent queue: Default is toggled off. When toggled on:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to the Cribl Stream data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from the Cribl Stream data processing engine.

`Smart` mode only engages when necessary, such as when a downstream Destination becomes blocked *and* the **Buffer size limit** reaches its limit. When persistent queue is set to `Smart` mode, Cribl attempts to flush the queue when every new event arrives. The only time events stay in the buffer is when a downstream Destination becomes blocked.

Buffer size limit: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. This buffer is for all connections, not just per Worker Process. For that reason, this can dramatically expand memory usage. Connections share this limit, which may result in slightly lower throughput for higher numbers of connections. For higher numbers of connections, consider increasing the limit.

Commit frequency: The number of events to send downstream before committing that Stream has read them. Defaults to `42`.

File size limit: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, and so forth. If not specified, Cribl Stream applies the default `1 MB`.

Queue size limit: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data. Required, and defaults to `5 GB`. Accepts positive numbers with units of `KB`, `MB`, `GB`, and so forth. Can be set as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** in Group or Fleet settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file closes. Defaults to `None`; `Gzip` is also available.



In Cribl Stream 4.1 and newer, the Source persistent queue default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Optimize Source Persistent Queues (sPQ).

You can optimize Workers' startup connections and CPU load at **Group/Fleet settings** > Worker Processes.

Processing Settings Fields

In this section, you can define new fields or modify existing ones using JavaScript expressions, similar to the Eval function.

- The **Field Name** can either be a new field (unique within the event) or an existing field name to modify its value.
- The **Value** is a JavaScript expression (enclosed in quotes or backticks) to compute the field's value (can be a constant). Select this field's advanced mode icon (far right) if you'd like to open a modal where you can work with sample data and iterate on results.

This flexibility means you can:

- Add new fields to enrich the event.
- Modify existing fields by overwriting their values.
- Compute logic or transformations using JavaScript expressions.

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline or Pack to process data from this input before the data is sent through the Routes.

Advanced Settings

Include varbind types: Toggle on to include varbind types in the `varbinds` field on each event. This is required for the SNMP Trap Serialize Function to serialize compliant events into SNMP traps before sending them to an SNMP Trap Destination. The field is parsed to an array of objects like, `[{oid: string, value: any, type: number}]`. Off by default.

IP allowlist regex: Regex matching IP addresses that are allowed to send data. Defaults to `.*`, that is, all IPs.

Buffer size limit (events): Maximum number of events to buffer when downstream is blocking. Defaults to `1000`.

UDP socket buffer size (bytes): Optionally, set the SO_RCVBUF socket option for the UDP socket. This value tells the operating system how many bytes can be buffered in the kernel before events are dropped. Leave blank to use the OS default. Min: `256`. Max: `4294967295`.

It may also be necessary to increase the size of the buffer available to the `SO_RCVBUF` socket option. Consult the documentation for your operating system for a specific procedure.



Setting this value will affect OS memory utilization.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

Internal Fields

Cribl Edge uses a set of internal fields to assist in handling of data. These "meta" fields are **not** part of an event, but they are accessible, and Functions can use them to make processing decisions.

Fields for this Source:

- `__didDecrypt`: Set to `true` for every trap that is successfully decrypted.
- `__final`
- `__inputId`
- `__snmpRaw`: Buffer containing Raw SNMP packet
- `__snmpVersion`: Acceptable values are `0`, `2` , or `3`. These respectively indicate SNMP v1, v2c, and v3.
- `__srcIpPort` : In this particular Source, this field uses a pipe (|) symbol to separate the source IP address and the port, in this format: `event.__srcIpPort = ${rInfo.address}|${rInfo.port};`
- `_time`

Considerations for Working with SNMP Trap Data

- It's possible to work with SNMP metadata (that is, we'll decode the packet). Options include dropping, routing, and so on.
- SNMP packets can be forwarded to other SNMP destinations. However, the contents of the incoming packet **cannot** be modified - that is, we'll forward the packets verbatim as they came in.
- SNMP packets can be forwarded to non-SNMP destinations (for example, Splunk, Syslog, S3, and so on).
- Non-SNMP input data **cannot** be sent to SNMP destinations.

Troubleshooting

The Source's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they are ingested. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the ingestion process, including any errors or warnings that may have occurred.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify ingestion issues. Analyze the graphs showing events and bytes in/out over time.

12.2.17. Syslog Source

The Syslog Source receives syslog data (UDP/TCP) from various devices. Syslog messages are parsed into structured fields or stored in a raw format if unrecognized. For high-volume scenarios, TCP load balancing distributes data, optimizing performance and minimizing CPU strain.

Compatible RFCs:

- Supports structured syslog data defined by RFC 3164 and RFC 5424, which can include timestamps, facility codes, severities, and message content for detailed analysis.
- Processes message-length prefixes specified in RFC 5425 and RFC 6587, ensuring complete and accurate data transmission.

Type: **Push** | TLS Support: **YES** | Event Breaker Support: **No**

For details on how to replace your syslog server with Cribl Edge, and for configuration examples, see these resources:

- Syslog Best Practices.
- Syslog to Cribl Stream Reference Architecture.
- Palo Alto Syslog to Cribl.Cloud.
- CEF/LEEF Syslog Pack.
- Exabeam Pack for Palo Alto Networks NGFW.

Looking for the Syslog Source in **Cribl Search**? See Ingest Syslog Messages into Cribl Search.

Configure Cribl Edge to Receive Data over Syslog

Cribl Edge ships with a Syslog Source preconfigured to listen for both UDP and TCP traffic on Port 9514. You can clone or directly modify this Source to further configure it, and then enable it.

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Source** and select the Source you want from the list, choosing either **Select Existing** or **Add New**.

- To configure via the Routes, select **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). Select the Source you want. Next, select **Add Source**.

2. In the **New Source** modal, configure the following under **General Settings**:
   - **Input ID**: Enter a unique name to identify this Syslog Source definition. The default Source is prefilled with the value `in_syslog_default`. which can't be changed via the UI. If you clone this Source, Cribl Edge will add `-CLONE` to the original **Input ID**.



The default filter expression from the **Input ID** field, `__inputId.startsWith('syslog:in_syslog:')` includes a trailing colon, which is omitted in the **Preview Full** mode of Data Preview. Therefore, when testing Routes and Pipelines, adjust your filter by removing the colon to match **Preview Full**. This issue is limited to testing and does not affect normal data processing.

   - **Description**: Optionally, enter a description.
   - **Address**: Enter the hostname/IP on which to listen for data. For example, `localhost` or `0.0.0.0`. For details, see IP Address Considerations.
   - **UDP port**: Enter the UDP port number to listen on. Not required if listening on TCP. The maximum supported inbound UDP message size is 16,384 bytes. For details, see UDP Port Considerations.
   - **TCP port**: Enter the TCP port number to listen on. Not required if listening on UDP.
3. Next, you can configure the following **Optional Settings**:
   - **Fields to keep**: List of fields from source data to retain and pass through. Supports wildcards. Defaults to `*` wildcard, meaning keep all fields. Fields **not** specified here (by wildcard or specific name) will be removed from the event.
   - **Tags**: Optionally, add tags that you can use to filter and group Sources in the Cribl Edge UI. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
4. Optionally, you can adjust the TLS, Persistent Queue, Processing and Advanced settings, or Connected Destinations outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

IP Address Considerations

Cribl recommends that you change the **Address** setting from its default `0.0.0.0` only if you know precisely why you're doing so. There are very few use cases where changing this broad default makes sense. All your Worker Nodes in each Worker Group receive this config, and no two Nodes should have the same IPs, other than `0.0.0.0` or `127.0.0.1`. The `0.0.0.0` tells the system to listen to all interfaces and addresses. A `127.0.0.1` entry means listen **only** on `localhost`.

UDP Port Considerations

Sending large numbers of UDP events per second can cause Cribl.Cloud to drop some of the data. This is due to the UDP protocol's restrictions. To minimize the risk of data loss, deploy a customer-managed (hybrid) Stream Worker Group with Worker Nodes as close as possible to the UDP senders. Cribl also recommends tuning the OS UDP buffer size.

Listening on Port 514

For Syslog devices that are hardcoded to send to port 514, you can listen on this port on Cribl.Cloud, which otherwise is normally available only with privileged access. To do this, enable the built-in `in_syslog_default` Syslog Source, which forwards traffic from port 514 to 10514.

Note that traffic on port 514 is unencrypted. This means that using this port may result in collecting fake or undesirable data. Enable the `in_syslog_default` Source only if you are aware of this risk.

TLS Settings (TCP Only)

On Cribl.Cloud, you can enable TLS using Cribl-provided certficates if desired.

Enabled: Defaults to toggled off. When toggled on:

Certificate: Name of the predefined certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

Passphrase: Passphrase to use to decrypt private key.

Certificate path: Server path containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

CA certificate path: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

Authenticate client (mutual auth): Require clients to present their certificates. Used to perform mutual authentication using SSL certs. Default is toggled off. When toggled on:

- **Validate client certificates**: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (for example, the system's CA). Default is toggled on.
- **Common name**: Regex that a peer certificate's subject attribute must match in order to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. (For example, to match the subject `CN=worker.cribl.local`, you would enter:

`worker\.cribl\.local`.) If the subject attribute contains Subject Alternative Name (SAN) entries, the Source will check the regex against all of those but ignore the Common Name (CN) entry (if any). If the certificate has no SAN extension, the Source will check the regex against the single name in the CN.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In the **Persistent Queue Settings** tab, you can optionally specify persistent queue storage, using the following controls. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Source Persistent Queues (sPQ)

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the **Enable persistent queue** toggle. If enabled, PQ is automatically configured in `Always On` mode, with a maximum queue size of 1 GB disk space allocated per PQ-enabled Source, per Worker Process.

The 1 GB limit is on uncompressed inbound data, and the queue does not perform any compression. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable persistent queue: Default is toggled off. When toggled on:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to the Cribl Stream data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from the Cribl Stream data processing engine.

`Smart` mode only engages when necessary, such as when a downstream Destination becomes blocked *and* the **Buffer size limit** reaches its limit. When persistent queue is set to `Smart` mode, Cribl attempts to flush the queue when every new event arrives. The only time events stay in the buffer is when a downstream Destination becomes blocked.

Buffer size limit: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. This buffer is for all connections, not just per Worker Process. For that reason, this can dramatically expand memory usage. Connections share this limit, which may result in slightly lower throughput for higher numbers of connections. For higher numbers of connections, consider increasing the limit.

Commit frequency: The number of events to send downstream before committing that Stream has read them. Defaults to `42`.

File size limit: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, and so forth. If not specified, Cribl Stream applies the default `1 MB`.

Queue size limit: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, and so forth. Can be set as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** in Group or Fleet settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file closes. Defaults to `None`; `Gzip` is also available.



In Cribl Stream 4.1 and newer, the Source persistent queue default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Optimize Source Persistent Queues (sPQ).

You can optimize Workers' startup connections and CPU load at **Group/Fleet settings** > Worker Processes.

Processing Settings

On this tab, you can enrich fields with added fields, and attach a pre-processing Pipeline to condition inbound syslog data.

Fields

In this section, you can define new fields or modify existing ones using JavaScript expressions, similar to the Eval function.

- The **Field Name** can either be a new field (unique within the event) or an existing field name to modify its value.

- The **Value** is a JavaScript expression (enclosed in quotes or backticks) to compute the field's value (can be a constant). Select this field's advanced mode icon (far right) if you'd like to open a modal where you can work with sample data and iterate on results.

This flexibility means you can:

- Add new fields to enrich the event.
- Modify existing fields by overwriting their values.
- Compute logic or transformations using JavaScript expressions.

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline or Pack to process data from this input before the data is sent through the Routes.

Advanced Settings

Enable proxy protocol: Toggle on if the connection is proxied by a device that supports PROXY protocol v1 or v2.

- **Enable enhanced TLS handshake for proxy protocol**: Toggle on to make TLS handshakes compatible with the PROXY protocol v1 and v2. When enabled, PROXY headers are parsed during the TLS handshake, reliably preserving the original source IP and port even for encrypted traffic routed through proxies or load balancers that prepend these headers.

Single msg per UDP: Toggle on to treat received UDP packet data as a full syslog message. If toggled off (default) Cribl Edge will treat newlines within the packet as event delimiters.

Allow non-standard app name: Toggle on to allow hyphens to appear in an RFC 3164-formatted Syslog message's `TAG` section. For details, see TAG Section Processing.

Enable TCP load balancing: Toggle on if you want the Source to load balance traffic across all Worker Processes, as explained in TCP Load Balancing. This setting is available only in Cribl Stream single-instance and Distributed deployments and when a **TCP port** is configured.

IP allowlist regex: Regex matching IP addresses that are permitted to send data. Defaults to `.*` (that is, all IPs).

Active connection limit: Maximum number of active connections allowed per Worker Process. Defaults to `1000`. Set a lower value if connection storms are causing the Source to hang. Set `0` for unlimited connections.

TCP socket idle timeout (seconds): The duration that Cribl Edge will wait for activity on an idle TCP socket before closing the connection. Disabled when set to `0`, the default.

TCP forced socket termination timeout (seconds): The extra time the server waits before forcibly closing a socket that has been idle (**TCP socket idle timeout**) or exceeded its maximum lifespan (**TCP socket max**

lifespan) but has not yet properly closed. This prevents resource leaks caused by unresponsive clients or network issues. Configure based on network latency and client behavior. Default: `30` seconds. Set to `0` to disable.

TCP socket max lifespan (seconds): The duration that a socket is allowed to remain open, regardless of activity. This setting prevents resource exhaustion (such as TCP pinning) by limiting the lifespan of connections. Configure based on expected connection durations and resource availability. Disabled when set to `0`, the default.

Buffer size limit (events): Maximum number of events to buffer when downstream is blocking. The buffer is only in memory. (This setting is applicable only to UDP syslog.) Events dropped because they exceed this buffer are logged as dropped in `stats` messages.

UDP socket buffer size (bytes): Optionally, set the SO_RCVBUF socket option for the UDP socket. This value tells the operating system how many bytes can be buffered in the kernel before events are dropped. Leave blank to use the OS default. Min: `256`. Max: `4294967295`.

You might also need to increase the size of the buffer available to the `SO_RCVBUF` socket option. Consult the documentation for your operating system for a specific procedure.



Setting this value will affect OS memory utilization.

Default timezone: Timezone to assign to timestamps that omit timezone info. Accept the default `Local` value, or use the drop-down list to select a specific timezone by city name or GMT/UTC offset.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

TCP Load Balancing



This feature is available for Cribl Stream deployments running in either single-instance or Distributed mode. The UI displays the setting when a **TCP port** is configured.

In single-instance mode, enabling TCP load balancing still starts a dedicated Load Balancer Worker Process. Ensure the host has enough CPU capacity to run the Load Balancer plus the configured Worker Processes.

When the Syslog Source receives a high volume of data over TCP, a single Worker Process can place excessive CPU load on its Cribl Stream Worker Node, while remaining Worker Processes on the same node mostly sit idle. This undesirable condition is called "TCP pinning" because the high volume of traffic "pins" a single TCP connection.

To alleviate TCP pinning, try toggling **Advanced Settings** > **Enable TCP load balancing** on. The Syslog Source will then load balance traffic across all Worker Processes.

When TCP load balancing is enabled, the Worker Node forks a special Worker Process dedicated to load balancing; creates sockets for communication between the load balancer and the regular Worker Processes; and, makes TCP load balancing metrics available.

The Load Balancing Process

The Worker Node forks a new, special **Load Balancer** Worker Process that distributes incoming syslog data among the other Worker Processes. For customer-managed (hybrid or on-prem) Fleets, consider reducing the Process count by 1, if possible. This will leave a core free to run the load balancer process.

If you want to verify that the load balancer Worker Process exists, teleport into the relevant Worker and in the **Worker Processes** tab you will see a Worker Process with `LB` in its name.

To view logs from the Leader UI, navigate to **Monitoring** > **Logs** > `<Worker Node ID>` > **Load Balancer**.

Worker Process Socket Files

The load balancer Worker Process sends data to the regular Worker Processes over inter-process communication (IPC) sockets. This means that in the `state` directory, there is a socket file for each load balancer Worker Process and each Worker Process.

By default, the socket files are linked to `state`, from the Cribl `tmp` directory. For customer-managed (hybrid or on-prem) Stream Worker Groups, you can specify a directory other than `tmp` for these links. Cribl recommends that you do **not** use this option, but if you find it necessary, navigate to **Worker Group/Fleet Settings** > **General Settings** > **Sockets** to configure as desired.

TCP Load Balancing Metrics

Metrics specific to TCP load balancing become available.

- `lb.bytes_out` - Total bytes processed by the load balancer Worker Process, by Source.
- `lb.writable_sockets` - Total unblocked Worker Process sockets, by Source.
- `lb.blocked_sockets` - Total blocked Worker Process sockets, by Source.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source data via the Routing table.

Select **QuickConnect** to send the data from this Source to one or more Destinations via independent, direct connections.

TAG Section Processing

Cribl Edge will try to parse `appname` out of a syslog message's `TAG` section, even when the message is not RFC 5424-formatted.

This practice means that the `TAG` section can contain any alphanumeric character or an underscore (`_`). When the **Allow-non-standard app name** option is enabled, Cribl Edge will also process hyphens that appear in an RFC 3164-formatted Syslog message's `TAG` section. If Cribl Edge encounters a hyphenated `appname`, it will continue processing to find `procid`. (This setting has no effect on RFC 5424-formatted messages.)

If the `TAG` section contains any non-alphanumeric character, Cribl Edge will treat that character as the termination of the `TAG` section, and as the starting character of the `CONTENT` section.

`CONTENT` Section Processing

Cribl Edge will try to parse `procid` from the beginning of the `CONTENT` section if this section is directly after the `TAG` section, and if it either follows a `:` or is surrounded by `[]`. This process occurs regardless of the **Allow-non-standard app name** setting.

What Fields to Expect

To create fields, the Syslog Source requires messages to comply with RFC 3164 or RFC 5424. If messages sent to the Source comply with neither of these RFCs, you'll see the following fields:

- `_raw`: Contains the whole message, but no other fields broken out.
- `_time`: Contains the time the message was received by Cribl Edge.
- `__srcIpPort <udp|tcp>:<port>`: See Internal Fields.
- `__syslogFail`: See Internal Fields.
- `host`: IP address of the device sending the incoming message.

If messages sent to the Source comply with either RFC 3164 or RFC 5424, fields that the RFC deems guaranteed will always be there, but fields deemed optional might or might not be. Once Cribl Edge parses the required fields and any optional fields, what remains is the actual message.

To see this in real life, install the `cribl-syslog-input` Pack and preview the `RFC5424-RFC3164.log` sample file.

RFC Name for Field	Cribl Name for Field	Guaranteed?	Notes
`PRI`	`facility`, `facilityName`, `severity`, `severityName`	Yes	`PRI` is a bitwise representation of Facility and Severity, which is how Cribl Edge can derive values for all four of its fields.
`TIMESTAMP`	`_time`	Yes	This field's format differs depending on which RFC the messages adhere to. Cribl Edge converts it to Unix time.
`HOSTNAME`	`host`	Yes	
`APP-NAME`	`appname`	No	
`PROCID`	`procid`	No	
`MSGID`	`msgid`	No	
`STRUCTURED-DATA`	`structuredData`	No	
`MSG`	`message`	Yes	

Event Size and Storage

By default, a Syslog Source produces eight parsed fields: `_time`, `appname`, `facility` (numeric), `facilityName` (text), `host`, `message`, `severity` (numeric), and `severityName` (text). This improves readability but does not reduce event volume. It also introduces redundancy, numeric and text pairs for `facility` and `severity`.

If you keep all of the default fields (the default behavior), events written to downstream storage are larger than the original raw syslog line because the original message is accompanied by additional fields. For short messages, this overhead can be significant relative to the raw size and could roughly double the stored size. For larger messages, the percentage increase is lower.

To control size growth, configure trimming at the Source or in a Pipeline:

- Adjust the Source **Fields to keep** setting to include only the fields you require. By default, all fields are kept.
- Use a Pipeline to remove redundant/unnecessary fields and, if appropriate, shrink outbound `_raw` to only the message component.

For a walk-through of optimization steps and expected results, see Syslog Best Practices: The Goal–Optimizing Syslog Events.

Internal Fields

Cribl Edge uses a set of internal fields to assist in handling of data. These "meta" fields are **not** part of an event, but are accessible and Functions can use them to make processing decisions.

Fields for this Source:

- `__inputId`
- `__srcIpPort <udp|tcp>:<port>`: Identifies the port of the sending socket.
- `__syslogFail: true` for data that fails RFC 3164/5424 validation as syslog format.

`stats` Log Message

In Cribl Edge 4.2.2 and newer, `stats` log messages report the number of events received, buffered, or dropped for exceeding the maximum Cribl buffer size. By default, these messages are logged every 60 seconds. These values also appear in **Sources** > **Syslog** > <Source_name> > **Status**, in the **UDP** row.

UDP Tuning

This section applies only to UDP-based syslog ingest with on-prem or hybrid Cribl Stream Worker Groups or Cribl Edge Fleets. It does not apply to TCP-based syslog, nor to Cribl-managed Stream Workers in Cribl.Cloud, where Cribl will manage UDP tuning for you.

Incoming UDP traffic is put into a buffer by the Linux kernel. Cribl will drain from that buffer as resources are available. At lower throughput levels, and with plenty of available processes, this isn't an issue. As you scale up, however, the default size of that buffer is very likely too small.

Monitoring Buffer Size and Packet Errors

You can check the current buffer size with the following command:

```
$ sysctl net.core.rmem_max
```

Copy

A typical value of about 200 KB is far too small for an even moderately busy syslog server.

You can check the health of UDP with the following command. Check the `packet receive errors` line.

```
$ netstat -su
```

Copy

If `packet receive errors` is more than zero, you have lost events, which is a particularly serious problem if the number of errors is increasing rapidly. This means you need to increase your `net.core.rmem_max` setting (covered later in this section).

Commands to Update Settings

You can update the live settings. But you'll also need to change the boot-time setting, so that next time you reboot, everything is ready to roll.

Live change, setting to 25 MB:

```
$ sysctl -w net.core.rmem_max=26214400
net.core.rmem_max = 26214400
$ sysctl -w net.core.rmem_default=26214400
net.core.rmem_default = 26214400
```

Copy

For the permanent settings change, add the following lines to `/etc/sysctl.conf`:

```
net.core.rmem_max=26214400
net.core.rmem_default=26214400
```

Copy

We recommend that you track a few things related to UDP receiving:

- The `netstat -su` command, watching for errors.
- The **Status** tab in the Syslog Source. In particular, watch for dropped messages. They could indicate you need a bigger buffer under **Advanced Settings** (default: 1000 events). They could also indicate your Worker is encountering pressure further down the Pipeline.
- Especially if you increase your kernel receive buffer as above, watch your Worker processes' memory usage.

Troubleshooting

The Source's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they are ingested. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the ingestion process, including any errors or warnings that may have occurred.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify ingestion issues. Analyze the graphs showing events and bytes in/out over time.



Cribl University offers an Advanced Troubleshooting > Source Integrations: Syslog short course. To follow the direct course link, first log into your Cribl University account. (To create an account, select the **Sign up** link. You'll need to review a short **Terms & Conditions** presentation, with chill music, before proceeding to courses - but Cribl's training is always free of charge.) Once logged in, check out other

12.2.18. TCP JSON Source

Cribl Edge can receive newline-delimited JSON data over TCP.

Type: **Push** | TLS Support: **YES** | Event Breaker Support: **No**

Looking for the TCP JSON Source in **Cribl Search**? See Ingest JSON Events via TCP into Cribl Search.

The TCP JSON Source is part of Cribl Edge's default configuration. If you delete it, Cribl Edge will re-create it when defaults are applied again (for example, when you create a new Fleet or deploy a configuration bundle). To prevent it from receiving data, disable it instead of deleting it.

Configure Cribl Edge to Receive TCP JSON Data

Cribl Edge ships with a TCP JSON Source preconfigured to listen on Port 10070. You can clone or directly modify this Source to further configure it, and then enable it.

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Source** and select the Source you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). Select the Source you want. Next, select **Add Source**.
2. Configure the following under **General Settings**:
    - **Input ID**: Enter a unique name. The default Source is prefilled with the value `in_tcp_json`, which can't be changed via the UI. If you clone this Source, Cribl Edge will add `-CLONE` to the original **Input ID**.
    - **Description**: Optionally, enter a description.
    - **Address**: Enter hostname/IP to listen for TCP JSON data. For example, `localhost` or `0.0.0.0`.
    - **Port**: Enter the port number to listen on.
3. Under **Authentication**, select an **Authentication method** from the dropdown:
    - **Manual**: Use this default option to enter the shared secret that clients must provide in the `authToken` header field. Exposes an **Auth token** field for this purpose. (If left blank, unauthorized

access will be permitted.) A **Generate** link is available if you need a new secret.

- **Secret**: This option exposes an **Auth token (text secret)** drop-down, in which you can select a stored secret that references the `authToken` header field value described above. The secret can reside in Cribl Edge's internal secrets manager or (if enabled) in an external KMS. A **Create** link is available if you need a new secret.

4. Next, you can configure the following **Optional Settings**:
   - **Tags**: Optionally, add tags that you can use to filter and group Sources in Cribl Edge's UI. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
5. Optionally, you can adjust the TLS, Persistent Queue Settings, Processing and Advanced settings, or Connected Destinations outlined in the sections below.
6. Select **Save**, then **Commit & Deploy**.

TLS Settings (Server Side)



On Cribl.Cloud, you can enable TLS using Cribl-provided certficates if desired.

Enabled: Defaults to toggled off. When toggled on:

Certificate: Name of the predefined certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

Passphrase: Passphrase to use to decrypt private key.

Certificate path: Server path containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

CA certificate path: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

Authenticate client (mutual auth): Require clients to present their certificates. Used to perform mutual authentication using SSL certs. Default is toggled off. When toggled on:

- **Validate client certificates**: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (for example, the system's CA). Default is toggled on.
- **Common name**: Regex that a peer certificate's subject attribute must match in order to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. (For example, to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.) If the subject attribute contains Subject Alternative Name (SAN) entries, the Source will check the regex against all of those but ignore the Common Name (CN) entry (if any). If the certificate has no SAN extension, the Source will check the regex against the single name in the CN.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In the **Persistent Queue Settings** tab, you can optionally specify persistent queue storage, using the following controls. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Source Persistent Queues (sPQ)

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the **Enable persistent queue** toggle. If enabled, PQ is automatically configured in `Always On` mode, with a maximum queue size of 1 GB disk space allocated per PQ-enabled Source, per Worker Process.

The 1 GB limit is on uncompressed inbound data, and the queue does not perform any compression. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable persistent queue: Default is toggled off. When toggled on:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to the Cribl Stream data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from the Cribl Stream data processing engine.



`Smart` mode only engages when necessary, such as when a downstream Destination becomes blocked *and* the **Buffer size limit** reaches its limit. When persistent queue is set to `Smart` mode, Cribl attempts to flush the queue when every new event arrives. The only time events stay in the buffer is when a downstream Destination becomes blocked.

Buffer size limit: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. This buffer is for all connections, not just per Worker Process. For that reason, this can dramatically expand memory usage. Connections share this limit, which may

result in slightly lower throughput for higher numbers of connections. For higher numbers of connections, consider increasing the limit.

Commit frequency: The number of events to send downstream before committing that Stream has read them. Defaults to `42`.

File size limit: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, and so forth. If not specified, Cribl Stream applies the default `1 MB`.

Queue size limit: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data. Required, and defaults to `5 GB`. Accepts positive numbers with units of `KB`, `MB`, `GB`, and so forth. Can be set as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** in Group or Fleet settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file closes. Defaults to `None`; `Gzip` is also available.



In Cribl Stream 4.1 and newer, the Source persistent queue default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Optimize Source Persistent Queues (sPQ).

You can optimize Workers' startup connections and CPU load at **Group/Fleet settings** > Worker Processes.

Processing Settings Fields

In this section, you can define new fields or modify existing ones using JavaScript expressions, similar to the Eval function.

- The **Field Name** can either be a new field (unique within the event) or an existing field name to modify its value.
- The **Value** is a JavaScript expression (enclosed in quotes or backticks) to compute the field's value (can be a constant). Select this field's advanced mode icon (far right) if you'd like to open a modal where you can work with sample data and iterate on results.

This flexibility means you can:

- Add new fields to enrich the event.
- Modify existing fields by overwriting their values.
- Compute logic or transformations using JavaScript expressions.

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline or Pack to process data from this input before the data is sent through the Routes.

Advanced Settings

Enable proxy protocol: Toggle on if the connection is proxied by a device that supports proxy protocol v1 or v2.

Enable load balancing (Cribl Stream only): Toggle on to distribute the data from incoming TCP connections across multiple Worker Processes. This improves performance compared to the default of having all data from an incoming connection processed on a single Worker Process.

- This spins up an extra Worker Process (named `wLB` in **Settings** > **Processes**) on the Worker Node, which handles splitting data from incoming TCP connections across all other Worker Processes.
- When enabled, the option to include common fields in the JSON header is not supported. This means fields specified under `fields` in the JSON header are not automatically added to all events.



Load balancing is available in Cribl Stream single-instance and Distributed deployments when a TCP port is configured.

IP allowlist regex: Regex matching IP addresses that are allowed to establish a connection. Defaults to `.*` (such as, all IPs).

Active connection limit: Maximum number of active connections allowed per Worker Process. Defaults to `1000`. Set a lower value if connection storms are causing the Source to hang. Set to `0` for unlimited connections.

Socket idle timeout (seconds): The duration that Cribl Edge will wait for activity on an idle TCP socket before closing the connection. Disabled when set to `0`, the default.

Forced socket termination timeout (seconds): The extra time the server waits before forcibly closing a socket that has been idle (**TCP socket idle timeout**) or exceeded its maximum lifespan (**TCP socket max lifespan**) but has not yet properly closed. This prevents resource leaks caused by unresponsive clients or network issues. Configure based on network latency and client behavior. Default: `30` seconds. Set to `0` to disable.

Socket max lifespan (seconds): The duration that a socket is allowed to remain open, regardless of activity. This setting prevents resource exhaustion (such as TCP pinning) by limiting the lifespan of connections. Configure based on expected connection durations and resource availability. Disabled when set to `0`, the default.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the configuration will be enabled everywhere.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

Internal Fields

Cribl Edge uses a set of internal fields to assist in handling of data. These "meta" fields are **not** part of an event, but they are accessible, and Functions can use them to make processing decisions.

Field for this Source:

- `__inputId`
- `__srcIpPort`

Format

Cribl Edge expects TCP JSON events in newline-delimited JSON format:

1. A header line. Can be empty - for example, `{}`. If **authToken** is enabled (see above) it should be included here as a field called `authToken`. When `authToken` is **not** set, the header line is **optional**. In this case, the first line will be treated as an event if does not look like a header record.
   In addition, if events need to contain common fields, they can be included here under `fields`. In the example below, `region` and `AZ` will be automatically added to all events.

   When you enable **Enable load balancing (Cribl Stream only)** under Advanced Settings, the Source will no longer support common fields in the JSON header. Consequently, any fields you define in the JSON header `fields` object will not be automatically propagated to each event.

2. A JSON event/record per line.

   Sample TCP JSON Events

```
{"authToken":"myToken42", "fields": {"region": "us-east-1", "AZ":"az1"}}

{"_raw":"this is a sample event ", "host":"myHost", "source":"mySource", "fieldA
{"host":"myOtherHost", "source":"myOtherSource", "_raw": "{\"message\":\"Somethi
```

Copy

TCP JSON Field Mapping to Splunk

If a TCP JSON Source is routed to a Splunk destination, fields within the JSON payload are mapped to Splunk fields. Fields that do not have corresponding (native) Splunk fields become index-time fields. For example, let's assume we have a TCP JSON event as below:

```
{"_time":1541280341, "host":"myHost", "source":"mySource", "_raw":"this is a sample event ", "fieldA":"valueA"}
```

Here, `_time`, `host`, and `source` become their corresponding fields in Splunk. The value of `_raw` becomes the actual body of the event, and `fieldA` becomes an index-time field (`fieldA::`valueA``).

Examples

Testing TCP JSON In

This first example simply tests that data is flowing in through the Source:

1. Configure Cribl Edge to listen on port `10001` for TCP JSON. Set `authToken` to `myToken42`.
2. Create a file called `test.json` with the payload above.
3. Send it over to your Cribl Edge host: `cat test.json | nc <myCriblHost> 10001`

Cribl Edge to Cribl.Cloud

This second example demonstrates using TCP JSON to send data from one Cribl Edge instance to a downstream Cribl.Cloud instance. We assume that the downstream Cloud instance uses Cribl.Cloud's **default** TCP JSON Source configuration.

So all the configuration happens on the upstream instance's TCP JSON Destination. Replace the `<organizationId>` placeholder with the Org ID from your Cribl.Cloud portal.

TCP JSON Destination Configuration

On the upstream Cribl Edge instance's Destination, set the following field values to match the target Cloud instance's defaults:

General Settings

Address: `default.main.<organizationId>.cribl.cloud` - you can simply copy/paste your Cribl.Cloud portal's **Ingest Endpoint** here. With a Cribl.Cloud Enterprise plan, generalize the `default.main` substring in this URL to `<groupName>.main` when sending to other Fleets.

Port: `10070`

TLS Settings (Client Side)

Enabled: Toggle on

Validate server certs: Toggle on

Periodic Logging

Cribl Edge logs metrics about incoming requests and ingested events once per minute.

These logs are stored in the `metrics.log` file. To view them in the UI, open the Source's **Logs** tab and choose **Worker Process X Metrics** from the drop-down, where **X** is the desired Worker Process.

This kind of periodic logging helps you determine whether a Source is in fact still healthy even when no data is coming in.

Troubleshooting

The Source's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they are ingested. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the ingestion process, including any errors or warnings that may have occurred.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify ingestion issues. Analyze the graphs showing events and bytes in/out over time.

12.2.19. TCP (Raw) Source

Cribl Edge supports receiving of data over TCP. (See examples and header options below.)



Type: **Push** | TLS Support: **YES** | Event Breaker Support: **YES**

Looking for the TCP (Raw) Source in **Cribl Search**? See Ingest Raw TCP Data into Cribl Search.

The TCP (Raw) Source is part of Cribl Edge's default configuration. If you delete it, Cribl Edge will re-create it when defaults are applied again (for example, when you create a new Fleet or deploy a configuration bundle). To prevent it from receiving data, disable it instead of deleting it.

Configure Cribl Edge to Receive TCP Data



Cribl Edge ships with a TCP Source preconfigured to listen on Port 10060. You can clone or directly modify this Source to further configure it, and then enable it.

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Source** and select the Source you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). Select the Source you want. Next, select **Add Source**.
2. Configure the following under **General Settings**:
    - **Input ID**: Enter a unique name to identify this TCP Source definition. If you clone this Source, Cribl Edge will add `-CLONE` to the original **Input ID**.
    - **Description**: Optionally, enter a description.
    - **Address**: Enter hostname/IP to listen for raw TCP data. For example, `localhost` or `0.0.0.0`.
    - **Port**: Enter port number.
    - **Enable header**: Toggle on to indicate that client will pass a header record with every new connection. The header can contain an `authToken`, and an object with a list of fields and values to add to every event. These fields can be used to simplify Event Breaker selection, routing, and so forth. Header format: `{ "authToken" : "myToken", "fields": { "field1": "value1", "field2": "value2" }}`.

- **Authentication method**: Select **Manual** to enter an auth token directly, or **Secret** to use a text secret to authenticate.
  - **Auth token**: If you selected **Manual**, enter an auth token, or select **Generate** to automatically generate a token. If empty, unauthorized access is permitted.
  - **Auth token (text secret)**: If you selected **Secret**, enter a shared secret to be provided by any client (in `authToken` header field). Select **Create** to create a new secret. If empty, unauthorized access will be permitted.

3. Next, you can configure the following **Optional Settings**:
   - **Tags**: Optionally, add tags that you can use to filter and group Sources in Cribl Edge's UI. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
4. Optionally, you can adjust the TLS Settings, Persistent Queue Settings, Processing, and Advanced settings, or Connected Destinations outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

TLS Settings (Server Side)



On Cribl.Cloud, you can enable TLS using Cribl-provided certficates if desired.

Enabled: Defaults to toggled off. When toggled on:

Certificate: Name of the predefined certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

Passphrase: Passphrase to use to decrypt private key.

Certificate path: Server path containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

CA certificate path: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

Authenticate client (mutual auth): Require clients to present their certificates. Used to perform mutual authentication using SSL certs. Default is toggled off. When toggled on:

- **Validate client certificates**: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (for example, the system's CA). Default is toggled on.
- **Common name**: Regex that a peer certificate's subject attribute must match in order to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. (For example, to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.) If the subject attribute contains Subject Alternative Name (SAN) entries, the

Source will check the regex against all of those but ignore the Common Name (CN) entry (if any). If the certificate has no SAN extension, the Source will check the regex against the single name in the CN.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In the **Persistent Queue Settings** tab, you can optionally specify persistent queue storage, using the following controls. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Source Persistent Queues (sPQ)

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the **Enable persistent queue** toggle. If enabled, PQ is automatically configured in `Always On` mode, with a maximum queue size of 1 GB disk space allocated per PQ-enabled Source, per Worker Process.

The 1 GB limit is on uncompressed inbound data, and the queue does not perform any compression. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable persistent queue: Default is toggled off. When toggled on:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to the Cribl Stream data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from the Cribl Stream data processing engine.

`Smart` mode only engages when necessary, such as when a downstream Destination becomes blocked *and* the **Buffer size limit** reaches its limit. When persistent queue is set to `Smart` mode, Cribl attempts to flush the queue when every new event arrives. The only time events stay in the buffer is when a downstream Destination becomes blocked.

Buffer size limit: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. This buffer is for all connections, not just per Worker Process. For that reason, this can dramatically expand memory usage. Connections share this limit, which may result in slightly lower throughput for higher numbers of connections. For higher numbers of connections, consider increasing the limit.

Commit frequency: The number of events to send downstream before committing that Stream has read them. Defaults to `42`.

File size limit: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, and so forth. If not specified, Cribl Stream applies the default `1 MB`.

Queue size limit: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, and so forth. Can be set as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** in Group or Fleet settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file closes. Defaults to `None`; `Gzip` is also available.



In Cribl Stream 4.1 and newer, the Source persistent queue default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Optimize Source Persistent Queues (sPQ).

You can optimize Workers' startup connections and CPU load at **Group/Fleet settings** > Worker Processes.

Processing Settings Custom Command

In this section, you can pass the data from this input to an external command for processing before the data continues downstream.

Enabled: Default is toggled off. When toggled on:

Command: Enter the command that will consume the data (via `stdin`) and will process its output (via `stdout`).

Arguments: Click **Add Argument** to add each argument for the command. You can drag arguments vertically to resequence them.

Event Breakers

Event Breaker rulesets: A list of event breaking rulesets that will be applied to the input data stream before the data is sent through the Routes. Defaults to `System Default Rule`.

Event Breaker buffer timeout: How long (in milliseconds) the Event Breaker will wait for new data to be sent to a specific channel, before flushing out the data stream, as-is, to the Routes. Minimum `10` ms, default `10000` (10 sec), maximum `43200000` (12 hours).

Fields

In this section, you can define new fields or modify existing ones using JavaScript expressions, similar to the Eval function.

- The **Field Name** can either be a new field (unique within the event) or an existing field name to modify its value.
- The **Value** is a JavaScript expression (enclosed in quotes or backticks) to compute the field's value (can be a constant). Select this field's advanced mode icon (far right) if you'd like to open a modal where you can work with sample data and iterate on results.

This flexibility means you can:

- Add new fields to enrich the event.
- Modify existing fields by overwriting their values.
- Compute logic or transformations using JavaScript expressions.

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline or Pack to process data from this input before the data is sent through the Routes.

Advanced Settings

Enable proxy protocol: Toggle on if the connection is proxied by a device that supports Proxy Protocol v1 or v2. When this setting is enabled, the `__srcIpPort` internal field will show the original source IP address and port. When it is disabled, the `__srcIpPort` field will show the IP address and port of the proxy that forwarded the connection.

IP allowlist regex: Regex matching IP addresses that are allowed to establish a connection. Defaults to `.*` (i.e,. all IPs).

Active connection limit: Maximum number of active connections allowed per Worker Process. Defaults to `1000`. Set a lower value if connection storms are causing the Source to hang. Set `0` for unlimited connections.

Socket idle timeout (seconds): The duration that Cribl Edge will wait for activity on an idle TCP socket before closing the connection. Disabled when set to `0`, the default.

Forced socket termination timeout (seconds): The extra time the server waits before forcibly closing a socket that has been idle (**TCP socket idle timeout**) or exceeded its maximum lifespan (**TCP socket max lifespan**) but has not yet properly closed. This prevents resource leaks caused by unresponsive clients or network issues. Configure based on network latency and client behavior. Default: `30` seconds. Set to `0` to disable.

Socket max lifespan (seconds): The duration that a socket is allowed to remain open, regardless of activity. This setting prevents resource exhaustion (such as TCP pinning) by limiting the lifespan of connections. Configure based on expected connection durations and resource availability. Disabled when set to `0`, the default.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

Internal Fields

Cribl Edge uses a set of internal fields to assist in handling of data. These "meta" fields are **not** part of an event, but they are accessible, and functions can use them to make processing decisions.

Fields accessible for this Source:

- `__inputId`
- `__srcIpPort`
- `__channel`

TCP Source Examples

Every new TCP connection may contain an **optional** header line, with an `authToken` and a list of fields and values to add to every event. To use the Cribl Edge Cloud sample, copy the `<token_value>` out of your Cribl Edge Cloud TCP Source.

Sample test.raw (on-prem) | Sample test.raw (Cribl.Cloud)

```
{"authToken":"myToken42", "fields": {"region": "us-east-1", "AZ":"az1"}}

this is event number 1
this is event number 2
```

Copy

Enabling the Example - Cribl Edge

1. Configure Cribl Edge to listen on port `7777` for raw TCP. Set `authToken` to `myToken42`.
2. Create a file called `test.raw`, with the payload above.
3. Send it over to your Cribl Edge host, using this command: `cat test.raw | nc <myCriblHost> 7777`

Enabling the Example - Cribl.Cloud

Use netcat with `--ssl` and `--ssl-verify`:

Command-line test

```
cat test.raw | nc --ssl --ssl-verify default.main.<organizationId>.cribl.cloud 10060
```

Copy

With a Cribl.Cloud Enterprise plan, generalize the above URL's `default.main` substring to `<groupName>.main` when sending to other Fleets.

Troubleshooting

The Source's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they are ingested. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the ingestion process, including any errors or warnings that may have occurred.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify ingestion issues. Analyze the graphs showing events and bytes in/out over time.

12.2.20. UDP (Raw) Source

The UDP (Raw) Source allows Cribl Edge to ingest unparsed, asynchronous data streams. Because UDP (User Datagram Protocol) is a connectionless protocol, this Source is primarily used for high-velocity telemetry where low latency is prioritized over guaranteed delivery or for data streams where the overhead of TCP handshakes would impact performance.

While the UDP (Raw) Source is highly flexible, it should be treated as a fallback for non-standard data. For common network telemetry needs, use Sources such as the Syslog, NetFlow & IPFIX, or SNMP Trap first.

For important information about using the UDP (Raw) Source more effectively, see:

- Better Practices
- UDP Tuning



Type: **Push** | TLS Support: **NO** | Event Breaker Support: **NO**

Configure Cribl Edge to Receive Raw UDP Data

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Source** and select the Source you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). Select the Source you want. Next, select **Add Source**.
2. In the **New Source** modal, configure the following under **General Settings**:
    - **Input ID**: Enter a unique name to identify this raw UDP Source definition.
    - **Description**: Optionally, enter a description.
    - **Address**: Enter the hostname/IP to listen for raw UDP data. For example: `localhost`, `0.0.0.0`, or `::`.
    - **Port**: Enter the port number.
3. Next, you can configure the following **Optional Settings**:
    - **Tags**: Optionally, add tags that you can use to filter and group Sources in Cribl Edge's UI. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

4. Optionally, you can adjust the Persistent Queue Settings, Processing, and Advanced settings, or Connected Destinations outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

> Sending large numbers of UDP events per second can cause Cribl.Cloud to drop some of the data. This results from restrictions of the UDP protocol.
>
> To minimize the risk of data loss, deploy a customer-managed (hybrid) Stream Worker Group with Worker Nodes as close as possible to the UDP sender. Cribl also recommends tuning the OS UDP buffer size.

Persistent Queue Settings

In the **Persistent Queue Settings** tab, you can optionally specify persistent queue storage, using the following controls. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Source Persistent Queues (sPQ)

> On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the **Enable persistent queue** toggle. If enabled, PQ is automatically configured in `Always On` mode, with a maximum queue size of 1 GB disk space allocated per PQ-enabled Source, per Worker Process.
>
> The 1 GB limit is on uncompressed inbound data, and the queue does not perform any compression. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable persistent queue: Default is toggled off. When toggled on:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to the Cribl Stream data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from the Cribl Stream data processing engine.

`Smart` mode only engages when necessary, such as when a downstream Destination becomes blocked *and* the **Buffer size limit** reaches its limit. When persistent queue is set to `Smart` mode, Cribl attempts to flush the queue when every new event arrives. The only time events stay in the buffer is when a downstream Destination becomes blocked.

Buffer size limit: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. This buffer is for all connections, not just per Worker Process. For that reason, this can dramatically expand memory usage. Connections share this limit, which may result in slightly lower throughput for higher numbers of connections. For higher numbers of connections, consider increasing the limit.

Commit frequency: The number of events to send downstream before committing that Stream has read them. Defaults to `42`.

File size limit: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, and so forth. If not specified, Cribl Stream applies the default `1 MB`.

Queue size limit: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, and so forth. Can be set as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** in Group or Fleet settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file closes. Defaults to `None`; `Gzip` is also available.

In Cribl Stream 4.1 and newer, the Source persistent queue default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Optimize Source Persistent Queues (sPQ).

You can optimize Workers' startup connections and CPU load at **Group/Fleet settings** > Worker Processes.

Processing Settings Fields

In this section, you can define new fields or modify existing ones using JavaScript expressions, similar to the Eval function.

- The **Field Name** can either be a new field (unique within the event) or an existing field name to modify its value.

- The **Value** is a JavaScript expression (enclosed in quotes or backticks) to compute the field's value (can be a constant). Select this field's advanced mode icon (far right) if you'd like to open a modal where you can work with sample data and iterate on results.

This flexibility means you can:

- Add new fields to enrich the event.
- Modify existing fields by overwriting their values.
- Compute logic or transformations using JavaScript expressions.

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline or Pack to process data from this input before the data is sent through the Routes.

Advanced Settings

Single msg per UDP: Toggle on if each UDP message should be treated as an independent event. Toggle off (default) if the message should be broken on newlines to create multiple events.

Ingest raw bytes: Toggle on to add a `__rawBytes` field to each event containing an array of the bytes received as the UDP message.

IP allowlist regex: Regex matching IP addresses that are allowed to establish a connection. Defaults to `.*` (i.e, all IPs).

Buffer size limit (events): Maximum number of events to buffer when downstream is blocking. The buffer is only in memory.

UDP socket buffer size (bytes): Optionally, set the SO_RCVBUF socket option for the UDP socket. This value tells the operating system how many bytes can be buffered in the kernel before events are dropped. Leave blank to use the OS default. Min: `256`. Max: `4294967295`.

It may also be necessary to increase the size of the buffer available to the `SO_RCVBUF` socket option. Consult the documentation for your operating system for a specific procedure.



Setting this value will affect OS memory utilization.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

What Fields to Expect

The Raw UDP Source does minimal processing of the incoming UDP messages to remain consistent with the internal Cribl event model. For each UDP message received on the socket, you can expect an event with the following fields:

- `_raw`: Contains the UTF-8 representation of the entire message received (if **Single msg per UDP** is toggled on), or of the given line that was split out of the message.
- `_time`: Timestamp in Unix time (seconds) at which the message was received by Cribl Edge.
- `source`: A string in the form `udp|<remote IP address>|<remote port>`, indicating the remote sender.
- `host`: The hostname of the machine running Cribl Edge that ingested this event.

Also, the internal fields listed below will be present.

Internal Fields

Cribl Edge uses a set of internal fields to assist in handling of data. These "meta" fields are **not** part of an event, but they are accessible, and functions can use them to make processing decisions.

Fields accessible for this Source:

- `__inputId`
- `__srcIpPort`
- `__rawBytes`: When **Ingest raw bytes** is toggled on, this field will be an array containing the bytes of the UDP message.

Better Practices

High-throughput UDP ingestion is uniquely sensitive to Linux kernel limitations. Unlike TCP, adding more Worker Processes to a single Edge Node does not linearly improve UDP performance. Under high stress (particularly with small, high-frequency events like Syslog), excessive Worker Process counts can cause kernel-level socket contention, visible as elevated `ksoftirqd` CPU usage.

Scale Horizontally

Instead of using a few extra-large Edge Nodes with high Worker Process counts, scale out using more Edge Nodes with a moderate number of Worker Processes. For many environments, a footprint of 2–3 Edge Nodes with fewer than 10 Worker Processes per Node provides a stable balance.

Use the minimum number of Worker Processes required to sustain your throughput. Increasing Worker Process counts beyond the point of saturation can lead to a disproportionate jump in CPU overhead.

Isolate UDP Traffic

Isolate high-volume UDP Sources into their own dedicated Fleet. Since TCP scales effectively with larger Edge Nodes and higher Worker Process counts, keeping TCP and UDP on separate Fleets allows you to tune each according to their specific scaling behaviors without one starving the other for kernel resources.

OS and Network Tuning

The bottleneck for UDP is often the OS kernel, not the Cribl Edge application logic.

- **Kernel Buffers:** Consider tuning `net.core.rmem_max` to increase the size of the receive buffers, which helps prevent packet loss during bursts.
- **Advanced NIC Tuning:** Features like RSS (Receive Side Scaling) or RPS (Receive Packet Steering) can help distribute interrupt load across CPU cores, though results vary by hardware and environment.



These limitations are driven by how the Linux kernel distributes UDP packets and handles interrupts. While OS-level tuning may improve performance, the most reliable strategy remains horizontal scaling across smaller, dedicated nodes.

UDP Tuning



For guidance on handling UDP data during backpressure events, see UDP Sources and Backpressure.

Incoming UDP traffic is put into a buffer by the Linux kernel. Cribl Edge will drain from that buffer as resources are available. At lower throughput levels, and with plenty of available processes, this isn't an issue. As you scale up, however, the default size of that buffer may be too small.

You can check the current buffer size with:

```
$ sysctl net.core.rmem_max
```

Copy

A typical value of about 200 KB is far too small for an even moderately busy Syslog server. You can check the health of UDP with the following command. Check the `packet receive errors` line.

```
$ netstat -su
```

Copy

If `packet receive errors` is more than zero, you have lost events, which is a particularly serious problem if the number of errors is increasing rapidly. This means youneed to increase your `net.core.rmem_max` setting (see earlier).

You can update the live settings, but you'll also need to change the boot-time setting so next time you reboot everything is ready to roll.

Live change, setting to 25 MB:

```
$ sysctl -w net.core.rmem_max=26214400
net.core.rmem_max = 26214400
$ sysctl -w net.core.rmem_default=26214400
net.core.rmem_default = 26214400
```

Copy

For the permanent settings change, add the following lines to `/etc/sysctl.conf`:

```
net.core.rmem_max=26214400
net.core.rmem_default=26214400
```

Copy

We recommend you track a few things related to UDP receiving:

- The `netstat -su` command, watching for errors.
- The **Status** tab in the UDP (Raw) Source. In particular, watch for dropped messages. They could indicate you need a bigger buffer under **Advanced Settings** (default: 1000 events). They could also indicate your Worker is encountering pressure further down the Pipeline.

- Especially if you increase your kernel receive buffer as above, watch your Worker processes' memory usage.

Troubleshooting

The Source's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they are ingested. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the ingestion process, including any errors or warnings that may have occurred.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify ingestion issues. Analyze the graphs showing events and bytes in/out over time.

12.2.21. Zscaler Cloud NSS Source

The Zscaler Cloud NSS Source receives log data over HTTP/S from Zscaler Cloud Nanolog Streaming Service (NSS).

If you're sending from a Zscaler NSS Virtual Machine, use a Syslog Source over TCP.



Type: **Push** | TLS Support: **YES** | Event Breaker Support: **YES**

This Source supports gzip-compressed inbound data when the `Content-Encoding: gzip` connection header is set.

Zscaler Cloud NSS Feed

When adding a Zcaler Cloud NSS feed, configure the following:

- **API URL**: The URL varies depending on your Cribl environment. Use the same port and HEC endpoint you define in your Zscaler Cloud NSS Source.
    - **Cribl.Cloud**: Enter the Workspace **Public Ingress Address** in the following format: `https://<groupName>.<workspaceName>.<organizationId>.cribl.cloud:<port><hec-endpoint>`
    - **On-prem**: Enter the IP address or FQDN for either your Cribl Edge instance, or the load balancer you're using with your Cribl Edge instances.
- **HTTP Headers**:
    - **Key 1**: Enter `Authorization`.
    - **Value 1**: Enter the Auth Token from your Cribl Source.
    - **Key 2**: Enter `Content-Encoding`.
    - **Value 2**: Enter `gzip`.

Configure a Zscaler Cloud NSS Source

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Source** and select the Source you want from the list, choosing either **Select Existing** or

Add New.

- To configure via the Routes, select **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). Select the Source you want. Next, select **Add Source**.

2. Configure the following under **General Settings**:
   - **Input ID**: Enter a unique name to identify this Source definition. If you clone this Source, Cribl Edge will add `-CLONE` to the original **Input ID**.
   - **Description**: Optionally, enter a description.

   The next three settings configure the location where the Source listens for incoming HTTP(S) data.
   - **Address**: Enter the hostname/IP. Such as: `localhost` or `0.0.0.0`. Supports IPv4 and IPv6 addresses.
   - **Port**: Enter the network port.
   - **HEC endpoint**: Enter the absolute path. Defaults to `/services/collector`.
3. Next, you can configure the following **Optional Settings**:
   - **Allowed Indexes**: List the values allowed in the HEC event index field. Allows wildcards. Leave blank to skip validation.
   - **Zscaler HEC Acks**: Whether to respond with a `200` HTTP status code acknowledging receipt of the `ACK` request. If toggled off (default), the Source responds with a `400` code to indicate `ACK` processing is disabled.
   - **Tags**: Optionally, add tags that you can use to filter and group Sources in Cribl Edge's UI. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
4. Optionally, you can enable TLS and persistent queue, then configure processing and advanced settings.
5. Create an Auth Token. You'll provide the token in an HTTP header with the key `Authorization`, see the Overview section for details.
6. Select **Save**, then **Commit & Deploy**.

TLS Settings (Server Side)

Enabled: Defaults to toggled off. When toggled on:

Certificate: Name of the predefined certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

Passphrase: Passphrase to use to decrypt private key.

Certificate path: Server path containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

CA certificate path: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

Authenticate client (mutual auth): Require clients to present their certificates. Used to perform mutual authentication using SSL certs. Default is toggled off. When toggled on:

- **Validate client certificates**: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (such as, the system's CA). Default is toggled off.
- **Common name**: Regex that a peer certificate's subject attribute must match in order to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. (For example, to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.) If the subject attribute contains Subject Alternative Name (SAN) entries, the Source will check the regex against all of those but ignore the Common Name (CN) entry (if any). If the certificate has no SAN extension, the Source will check the regex against the single name in the CN.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In the **Persistent Queue Settings** tab, you can optionally specify persistent queue storage, using the following controls. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Source Persistent Queues (sPQ)

> On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the **Enable persistent queue** toggle. If enabled, PQ is automatically configured in `Always On` mode, with a maximum queue size of 1 GB disk space allocated per PQ-enabled Source, per Worker Process.
>
> The 1 GB limit is on uncompressed inbound data, and the queue does not perform any compression. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable persistent queue: Default is toggled off. When toggled on:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to the Cribl Stream data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from the Cribl Stream data processing engine.



`Smart` mode only engages when necessary, such as when a downstream Destination becomes blocked *and* the **Buffer size limit** reaches its limit. When persistent queue is set to `Smart` mode, Cribl attempts to flush the queue when every new event arrives. The only time events stay in the buffer is when a downstream Destination becomes blocked.

Buffer size limit: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. This buffer is for all connections, not just per Worker Process. For that reason, this can dramatically expand memory usage. Connections share this limit, which may result in slightly lower throughput for higher numbers of connections. For higher numbers of connections, consider increasing the limit.

Commit frequency: The number of events to send downstream before committing that Stream has read them. Defaults to `42`.

File size limit: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, and so forth. If not specified, Cribl Stream applies the default `1 MB`.

Queue size limit: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data. Required, and defaults to `5 GB`. Accepts positive numbers with units of `KB`, `MB`, `GB`, and so forth. Can be set as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** in Group or Fleet settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file closes. Defaults to `None`; `Gzip` is also available.



In Cribl Stream 4.1 and newer, the Source persistent queue default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Optimize Source Persistent Queues (sPQ).

You can optimize Workers' startup connections and CPU load at **Group/Fleet settings** > Worker Processes.

Processing Settings Fields

In this section, you can add Fields to each event using Eval-like functionality.

Name: Field name.

Value: JavaScript expression to determine field's value (can be a constant).



Fields specified on the **Fields** tab will normally override fields of the same name in events. But you can specify that fields in events should override these fields' values.

In particular, where incoming events have no `index` field, this Source adds one with the literal value `default`. You can override this value by using **Add Field** to specify an `index` field, and then setting its **Value** to an expression of the following form: `index == 'default' ? 'myIndex' : index`

Fields here are evaluated and applied **after** any fields specified in the Auth Tokens section.

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline or Pack to process data from this input before the data is sent through the Routes.

Auth Tokens

Provides the authentication token required to access this Source's data. Ensure the token is valid and has the necessary permissions for data retrieval.



Not providing a token allows **unauthorized access**.

Authentication method: Choose between manually entering a **Token** or selecting a stored **Secret**. If you opt for manual entry, type the token directly or **Generate** a new one. For added security, use the secrets store to select a stored **Token (text secret)** or **Create** a new one.

Enable token: Controls the status of the specified authentication token. When toggled on, the specified auth token is active and can be used for client access. If toggled off, the token is disabled, causing it to be skipped and ignored during initialization. This means that any client attempting to use a disabled token will not be granted access.

Disabling a token can be helpful when you need to temporarily stop receiving data from a particular service or application. Disabling a token prevents the service or application from sending data to your endpoint without having to completely delete the token.

See also Periodic Logging for information on how auth tokens affect product logging.

Description: Optional (but recommended), the description for this token. Although this field is optional, the token description becomes the display name for the token. You can select this display name to expand or collapse the token details.

Allowed indexes: Optionally, specify which indexes are permissible for events ingested using this token. Supports multiple index values and wildcards.

Fields: Fields to add to events referencing this token. Each field is a **Name/Value** pair.



Fields specified on the **Auth Tokens** tab will normally override fields of the same name in events. But you can specify that fields in events should override these fields' values.

In particular, where incoming events have no `index` field, this Source adds one with the literal value `default`. You can override this value by using **Add Field** to specify an `index` field, and then setting its **Value** to an expression of the following form: `index == 'default' ? 'myIndex' : index`

Fields here are evaluated and applied **before** any fields specified in the Fields section.

Advanced Settings

Emit per-token request metrics: Toggle on to emit per-token `<prefix>.http.perToken` and summary `<prefix>.http.summary` request metrics. This enables Periodic Logging.

Show originating IP: Toggle on when clients are connecting through a proxy that supports the `X-Forwarded-For` header to keep the client's original IP address on the event instead of the proxy's IP address. This setting affects how the Source handles the `__srcIpPort` field.

Capture request headers: Toggle on to add request headers to events, in the `__headers` field.

CORS allowed origins: If you need to enable cross-origin resource sharing with Zscaler senders, use this field to specify the HTTP origins to which Cribl Edge should send `Access-Control-Allow-*` headers. You can enter multiple domains and use wildcards. This and its companion **CORS allowed headers** option should seldom be needed - see Working with CORS Headers below.

CORS allowed headers: List HTTP headers that Cribl Edge will send in a CORS preflight response to your configured (above) **CORS allowed origins** as `Access-Control-Allow-Headers`. Enter `*` to allow all headers.

Active request limit (on-prem only): Maximum number of active requests allowed for this Source, per Worker Process. Defaults to `256`. Enter `0` for unlimited.

Raising this limit can increase throughput by allowing more concurrent data requests, but increases resource usage and load on both your Cribl Edge infrastructure and on downstream Destinations. Before raising this limit, ensure:

- Your Cribl Edge deployment has sufficient capacity to support higher request concurrency, including CPU, memory, or number of Worker Processes.
- Downstream Destinations are correctly sized and tuned to accept the higher data ingest rate, preventing backpressure. See Manage Backpressure for more information.

Improper sizing on either side can result in dropped events, delayed processing, or overall system instability.

Activity log sample rate: Determines how often request activity is logged at the `info` level. The default `100` value logs every 100th value; a `1` value would log every request; a `10` value would log every 10th request; and so forth.

Requests-per-socket limit: The maximum number of requests Cribl Edge should allow on one socket before instructing the client to close the connection. Defaults to `0` (unlimited). See Balancing Connection Reuse Against Request Distribution below.

Socket timeout (seconds): How long Cribl Edge should wait before assuming that an inactive socket has timed out. The default `0` value means wait forever.

Request timeout (seconds): How long to wait for an incoming request to complete before aborting it. The default `0` value means wait indefinitely.

Keep-alive timeout (seconds): After the last response is sent, Cribl Edge will wait this long for additional data before closing the socket connection. Defaults to `5` seconds; minimum is `1` second; maximum is `600` seconds (10 minutes).

The longer the **Keep-alive timeout**, the more Cribl Edge will reuse connections. The shorter the timeout, the closer Cribl Edge gets to creating a new connection for every request. When request frequency is high, you can use longer timeouts to reduce the number of connections created, which mitigates the associated cost.

IP allowlist regex: Grants access to requests originating from specific IP addresses that match a defined pattern. Unmatched requests are rejected with a 403 (Forbidden) status code. Defaults to `.*` (allow all).

IP denylist regex: Blocks requests originating from specific IP addresses that match a defined pattern, even if they would be allowed by default. Rejected requests receive a 403 (Forbidden) status code. Defaults to `^$` (allow all).

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Balancing Connection Reuse Against Request Distribution

Requests-per-socket limit allows you to limit the number of HTTP requests an upstream client can send on one network connection. Once the limit is reached, Cribl Edge uses HTTP headers to inform the client that it must establish a new connection to send any more requests. (Specifically, Cribl Edge sets the HTTP `Connection` header to `close`.) After that, if the client disregards what Cribl Edge has asked it to do and tries to send another HTTP request over the existing connection, Cribl Edge will respond with an HTTP status code of `503 Service Unavailable`.

Use this setting to strike a balance between connection reuse by the client, and distribution of requests among one or more Edge Node processes by Cribl Edge:

- When a client sends a sequence of requests on the same connection, that is called connection reuse. Because connection reuse benefits client performance by avoiding the overhead of creating new connections, clients have an incentive to **maximize** connection reuse.
- Meanwhile, a single process on that Edge Node will handle all the requests of a single network connection, for the lifetime of the connection. When receiving a large overall set of data, Cribl Edge performs better when the workload is distributed across multiple Edge Node processes. In that situation, it makes sense to **limit** connection reuse.

There is no one-size-fits-all solution, because of variation in the size of the payload a client sends with a request and in the number of requests a client wants to send in one sequence. Start by estimating how long connections will stay open. To do this, multiply the typical time that requests take to process (based on payload size) times the number of requests the client typically wants to send.

If the result is 60 seconds or longer, set **Requests-per-socket limit** to force the client to create a new connection sooner. This way, more data can be spread over more Edge Node processes within a given unit of time.

For example: Suppose a client tries to send thousands of requests over a very few connections that stay open for hours on end. By setting a relatively low **Requests-per-socket limit**, you can ensure that the same work is done over more, shorter-lived connections distributed between more Edge Node processes, yielding better performance from Cribl Edge.

A final point to consider is that one Cribl Edge Source can receive requests from more than one client, making it more complicated to determine an optimal value for **Requests-per-socket limit**.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

Internal Fields

Cribl Edge uses a set of internal fields to assist in handling of data. These "meta" fields are **not** part of an event, but they are accessible, and Functions can use them to make processing decisions.

Fields for this Source:

- `__headers` - Added only when **Advanced Settings** > **Capture request headers** is toggled on.
- `__hecToken`
- `__inputId`
- `__ctrlFields`
- `__srcIpPort` - See details below.
- `__TZ`

Overriding `__srcIpPort` with Client IP/Port

The `__srcIpPort` field's value contains the IP address and (optionally) port of the Zscaler Cloud NSS sending data to this Source.

When any proxies (including load balancers) lie between the Zscaler Cloud NSS and the Source, the last proxy adds an `X-Forwarded-For` header whose value is the IP/port of the original client. With multiple proxies, this header's value will be an array, whose first item is the original client IP/port.

If `X-Forwarded-For` is present, and **Advanced Settings** > **Show originating IP** is toggled off, the original client IP/port in this header will override the value of `__srcIpPort`.

If **Show originating IP** is toggled on, the `X-Forwarded-For` header's contents will **not** override the `__srcIpPort` value. (Here, the upstream proxy can convey the client IP/port without using this header.)

Working with CORS Headers

The **CORS allowed origins** and **CORS allowed headers** settings are needed **only** when you want JavaScript code running in a web browser to send events to the Zscaler Cloud NSS Source. In this (uncommon) scenario, the

code in question sends a preflight request that includes CORS (Cross-Origin Resource Sharing) headers, and the Zscaler Cloud NSS Source includes CORS headers in its response.

If the settings (and thus the CORS headers in the response) are correct, the browser will allow the code in question to complete requests to the Zscaler Cloud NSS Source - that is, allow the code to read the Source's responses. (These settings are optional because for some web applications it's sufficient to send requests **without** reading responses.)

The CORS header dance begins when the web browser sends a request with a header stating its origin. What happens next depends on how you've configured the CORS header settings, as explained in the tables below. (For these examples, we'll assume an origin of `https://mycoolwebapp:3001`.)

CORS allowed origins value	What Zscaler Cloud NSS Source does	Does Browser then allow code to read response?
`*`	Send `Access-Control-Allow-Origin: *` header.	Yes
Multiple wildcard values	If origin value matches any wildcard, send `Access-Control-Allow-Origin: https://mycoolwebapp:3001` header.	Yes
Nothing (not configured)	Send neither an `Access-Control-Allow-Origin` nor an `Access-Control-Allow-Headers` header.	No

The web browser's request might also include an `Access-Control-Request-Headers` header that lists one or more headers that it wants to use in a subsequent `POST` request (that is, the "actual" as opposed to preflight request). The Zscaler Cloud NSS Source can then respond with an `Access-Control-Allow-Headers` header **if and only if** it's also sending the `Access-Control-Allow-Origin` header.

CORS allowed headers value	What Zscaler Cloud NSS Source does	Does Browser then allow code to read response?
`*`	Send `Access-Control-Allow-Headers` header with the same list the client sent.	Yes
Multiple wildcard values	If any headers from the client's list match wildcards, send `Access-Control-Allow-Headers` header listing the matching headers.	Yes
Nothing (not configured)	Do not send an `Access-Control-Allow-Headers` header.	Probably not

Periodic Logging

Cribl Edge emits metrics about incoming requests when **Advanced Settings > Emit per-token request metrics** is enabled. These metrics can be routed to a Destination by enabling the Cribl Metrics Source. There are two categories of metrics: per-token metrics provide details for each configured token, while summary metrics aggregate data across all tokens. If no tokens are configured, only summary metrics will be emitted.

Per-token metric names will be prefixed by `<basePrefix>.http.perToken.`, where `<basePrefix>` is the configured base prefix for the Cribl Metrics Source. The available metric names are listed below, excluding the prefix. As these are counter-type metrics, they will not be emitted if the value is zero.

- `bytesReceived`
- `numRequests`
- `numEvents`
- `numErrors` *Note: this is cumulative of specific error types below as well as a catch-all for other types*
- `numParserErrors`
- `numToDisabledToken`

The summary metric names will be prefixed by `<basePrefix>.http.summary.` where `<basePrefix>` is the configured base prefix for the Cribl Metrics Source. The available metric names are listed below, excluding the prefix. As these are counter-type metrics, they will not be emitted if the value is zero.

- `bytesReceived`
- `numRequests`
- `numEvents`
- `numErrors` *Note: this is cumulative of specific error types below as well as a catch-all for other types*
- `numParserErrors`
- `numToDisabledToken`
- `numAuthFailures`
- `numToWrongUrl`
- `numAckReqs`
- `numReqsAcked`

Troubleshooting

The Source's configuration modal has helpful tabs for troubleshooting:

12.3. Destinations

Cribl Edge can send transformed data to various Destinations, including Cribl HTTP, Cribl TCP, Elasticsearch, Amazon Kinesis, Amazon S3 and other object stores, Prometheus and compatible services, InfluxDB, Splunk, Snowflake, Databricks, TCP JSON, and many others.

Destinations can write data to either IPv4 or IPv6 addresses.

Destination are grouped into categories that define how they handle unreachable outputs (backpressure events) and their load-balancing capabilities.

Destination Categories

Destinations can be divided into streaming and non-streaming: those that accept events in real time, and those that batch them from a staging directory.

Other than that, each Destination can belong to one of the following (non-exclusive) categories:

- Filesystem-based
- Load-balanced

Streaming and Non-Streaming Destinations

Streaming and non-streaming Destination differ in the way they receive events:

- Streaming Destinations accept events in real time.
- Non-streaming Destinations receive events in batches from a staging directory.

Non-Streaming Destinations

Non-streaming Destinations make use of the staging directory, and have specific behavior regarding batching events into files.

Staging Directory

With non-streaming Destinations, Cribl Edge uses a staging directory in the local filesystem to format and write outputted events before sending them to configured Destinations. After a set of conditions is met, data is compressed and then moved to the final Destination.

To reduce costs when the staging directory is also the final directory, Cribl Edge avoids iterating through all the files within a directory by keeping an inventory of open (in progress) files in the staging directory's root. At startup, Cribl Edge will check for any leftover files in progress from prior sessions, and will ensure that they're moved to their final Destination. The process of moving to the final Destination is delayed after startup (default delay: 30 seconds). Processing of these files is paced at one file per service period (which defaults to 1 second).

Batching Conditions

In non-streaming delivery, a file is closed and rolled out when it reaches its configured maximum:

- Size
- Open time
- Idle time

If a new file needs to be open, Cribl Edge will enforce the maximum number of open files by closing files in the order in which they were opened.

Filesystem-based Destinations

Some Destinations are Filesystem-based, which means they receive files on disk from a staging directory and batch them in a queue. When a batch of events is ready for transmission, Cribl Edge closes the file, optionally compresses it, and transmits the file to the downstream service. Filesystem-based Destinations do not support persistent queues.

Load-balanced Destinations

Certain Destinations offer built-in load-balancing capabilities.

Available Destinations

Cribl Edge supports the following Destinations. You can configure proxy servers for all HTTP-based Destinations.

Destination	Protocol	Streaming	Filesystem-Based	Load-Balanced
Amazon S3 Compatible Stores	HTTP/S	Non-streaming		
Amazon CloudWatch Logs	HTTP/S	Non-streaming		
Data Lakes > Amazon Security Lake	HTTP/S	Non-Streaming		

Destination	Protocol	Streaming	Filesystem-Based	Load-Balanced
Amazon Kinesis Data Streams	HTTP/S	Streaming		
Amazon MSK	TCP	Streaming		
Amazon SQS	HTTP/S	Streaming		
Azure Blob Storage	HTTPS only	Non-Streaming		
Azure Data Explorer	HTTPS only	Streaming or non-streaming		
Azure Event Hubs	TCP	Streaming		
Azure Monitor Logs	HTTPS only	Streaming		
Microsoft Sentinel	HTTP/S	Streaming		
ClickHouse	HTTP/S	Streaming		
Cloudflare R2	HTTP/S	Non-Streaming		
Cortex XSIAM	HTTP/S	Streaming		
Confluent Cloud	TCP	Streaming		
CrowdStrike Falcon LogScale	HTTPS only	Streaming		
CrowdStrike Falcon Next-Gen SIEM	HTTPS only	Streaming		
Datadog	HTTPS only	Streaming		
Databricks		Non-Streaming		
Dynatrace HTTP	HTTP/S	Streaming		
Dynatrace OTLP	HTTP/S	Streaming		
Elastic Cloud	HTTPS only	Streaming		
Elasticsearch	HTTP/S	Streaming		
Exabeam	HTTP/S	Non-Streaming		
Filesystem/NFS		Non-Streaming		

Destination	Protocol	Streaming	Filesystem-Based	Load-Balanced
Google Cloud Chronicle API	HTTPS only	Streaming		
Google Cloud Logging	HTTPS only	Streaming		
Google Cloud Pub/Sub	HTTPS only	Streaming		
Google Cloud Storage	HTTPS only	Non-Streaming		
Google SecOps	HTTPS only	Streaming		
Grafana Cloud	HTTP/S	Streaming		
Graphite	TCP or UDP	Streaming		
Honeycomb	HTTPS only	Streaming		
InfluxDB	HTTP/S	Streaming		
Kafka	TCP	Streaming		
Loki	HTTP/S	Streaming		
Microsoft Fabric Real-Time Intelligence	TCP	Streaming		
MinIO	HTTP/S	Non-Streaming		
New Relic Events	HTTPS only	Streaming		
New Relic Logs & Metrics	HTTPS only	Streaming		
OpenTelemetry (OTel)	gRPC or HTTP/S	Streaming		
Prometheus	HTTP/S	Streaming		
SentinelOne AI SIEM	HTTP/S	Streaming		
SentinelOne DataSet	HTTPS only	Streaming		
ServiceNow Cloud Observability	gRPC or HTTP/S	Streaming		
SignalFx	HTTPS only	Streaming		
SNMP Trap	UDP	Streaming		

Destination	Protocol	Streaming	Filesystem-Based	Load-Balanced
Splunk HEC	HTTP/S	Streaming		
Splunk Load Balanced	TCP	Streaming		
Splunk Single Instance	TCP	Streaming		
StatsD	TCP or UDP	Streaming		
StatsD Extended	TCP or UDP	Streaming		
Sumo Logic	HTTP/S	Streaming		
Syslog	TCP or UDP	Streaming		(TCP only)
TCP JSON	TCP	Streaming		
Wavefront	HTTPS only	Streaming		
Webhook	HTTP/S	Streaming		



You can adapt the Amazon S3 Compatible Stores Destination to send data to downstream services like Databricks and Snowflake, for which Cribl Edge currently has no preconfigured Destination. For details, please contact Cribl Support.

Internal Destinations

Internal Destinations are special-purpose Destinations that route data within your Cribl Edge deployment, or among Edge Nodes across distributed or hybrid Cribl.Cloud deployments. The following internal Destinations are available:

- Default: Specify a default output from among your configured Destinations.
- Output Router: A "meta-Destination." Configure rules that route data to multiple configured Destinations.
- DevNull: Simply drops events. Preconfigured and active when you install Cribl Edge, so it requires no configuration. Useful for testing.
- Cribl HTTP: Send data among peer Edge Nodes over HTTP. Streaming and load-balanced.
- Cribl TCP: Send data among peer Edge Nodes over TCP. Streaming and load-balanced.
- Cribl Search: Send data from Edge Fleets directly to Cribl Search over HTTP. Streaming and load-balanced.
- **SpaceOut**: This experimental Destination is undocumented. Be careful!

Connecting to Cribl Stream

Always use Cribl TCP or Cribl HTTP as the Destinations to send data to Cribl Stream. Otherwise, you might see duplication of certain data (such as host). Also, data sent to Cribl Stream via TCP/HTTP is not counted against your ingest data quota.

See Cribl Edge to Cribl Stream for more information.

Data Delivery to Unreachable Destinations

Cribl Edge attempts to deliver data to all Destinations that are configured to receive it at least once. When a Destination is unreachable, there are three possible behaviors:

- **Block** - Cribl Edge will block incoming events.
- **Drop** - Cribl Edge will drop events addressed to that Destination.
- **Queue** - To prevent data loss, Cribl Edge will write events to a **persistent queue** disk buffer, then forward them when a Destination becomes available. (Available on several streaming Destinations.)

For further information about backpressure (a situation when a Destination receives more data than it can send), see Destination Backpressure Triggers.

You can configure your desired behavior through a Destination's **Backpressure Behavior** drop-down. Where other options are not displayed, Cribl Edge's default behavior is **Block**. For details about all the above behaviors and options, see Persistent Queues.

To learn how backpressure affects upstream Sources when persistent queue is not enabled, see Backpressure Impacts to Sources.

How Destinations Count Bytes Sent

Cribl Destinations record bytes sent per event by setting an internal field, `__outLen`, at the point where the Destination serializes or writes the payload for delivery. The calculation varies by Destination type. `__outLen` reflects uncompressed bytes. Compression is applied after `__outLen` is set, so metrics represent logical (uncompressed) size rather than on-the-wire compressed size.

Some Destinations can operate in multiple transport/mode categories. For example, Azure Data Explorer supports both streaming and batching, and OpenTelemetry and Dynatrace OTLP support HTTP and gRPC. These Destinations have clarifications in parenthesis in the `Destinations` column of the Byte Calculations table.

Byte Calculations

Destination Category	Byte Calculations	Destinations
HTTP and API-based Destinations	Measures the size of the data payload sent in the HTTP request (the uncompressed event data), calculated either event-by-event or during batch creation.	Amazon CloudWatch Logs, Azure Data Explorer (streaming mode), CrowdStrike Next-Gen SIEM, Cribl HTTP, Dynatrace (HTTP), Dynatrace (OTLP HTTP), Elastic, Elastic Cloud, Exabeam, Grafana Cloud, Google Cloud Logging, Google Pub/Sub, Honeycomb, Humio (HEC), InfluxDB, Microsoft Fabric, Microsoft Sentinel, New Relic (Events), OpenTelemetry (HTTP), SentinelOne AI SIEM, ServiceNow, Splunk HEC, Subscription, Wavefront
Batched HTTP serialization	Measures the event size during batch assembly. The final byte count is assigned from the size of the event's data within the batched request body.	Azure Monitor Logs, ClickHouse, Datadog, DataSet, Google Cloud Chronicle, Grafana Loki, Palo Alto Cortex XSIAM, Sumo Logic, Webhook
Object storage	Measures the size of the content before compression and writing to the storage medium.	Amazon S3, Amazon Security Lake, Azure Blob Storage, Azure Data Explorer (batching mode), Cloudflare R2, Cribl Lake, Databricks, Disk Spool, DL S3, Filesystem, Google Cloud Storage, MinIO, Ring
TCP-based outputs	Measures the actual bytes written to the TCP network buffer for each event.	Cribl TCP, Graphite, Syslog (TCP), TCP JSON
Splunk Forwarder (S2S)	Measures the raw byte size for data events. Metric events are counted as a fixed 150 bytes each.	Splunk (Load Balanced), Splunk (S2S)
OpenTelemetry (OTLP/gRPC)	Measures the size of the data after it has been formatted (serialized) into the protobuf structure for gRPC transmission.	Dynatrace (OTLP gRPC), OpenTelemetry (OTLP gRPC)
Streaming and message queues	Measures the total size of the message, including the key, value (payload), and any headers.	Amazon Kinesis Data Streams, Amazon MSK, Amazon SQS, Apache Kafka, Azure Event Hubs, Confluent Cloud
Metrics line senders	Measures the byte size of the formatted metric line	StatsD Extended, StatsD

Destination Category	Byte Calculations	Destinations
	at the moment it is prepared for output.	
	Measures the size based on the metric components	

12.3.1. Manage Destinations

For each Destination **type**, you can create multiple definitions, each with different settings, to handle delivering data to different outputs with differing requirements.

Once a Destination definition is created, enabled and set up to receive data, you can verify that data is flowing correctly, and monitor the Destination status to check for issues.

Create a Destination

You can create a new Destination in two ways: by using QuickConnect or Routing.

For configuration settings specific to each Destination, check that Destination's documentation.

Create a Destination with QuickConnect

To create a new Destination via QuickConnect:

1. If you are on a Distributed deployment, first select **Fleets** in the sidebar and choose a Fleet.
2. Select **Collect**.
3. Select **Add Destination** at right, and select the Destination you want from the list.

For configuration settings specific to each Destination, check that Destination's documentation. After you've set up the Destination, you can verify its data flow.

Create a Destination with Routes

To create a new Destination via Routing:

1. If you are on a Distributed deployment, first select **Fleets** in the sidebar and choose a Fleet.
2. Select **More**, and then **Destinations**.
3. Select the Destination you want.
4. Select **Add Destination** to open a **New Destination** modal.

For configuration settings specific to each Destination, check that Destination's documentation. After you've set up the Destination, you can verify its data flow.

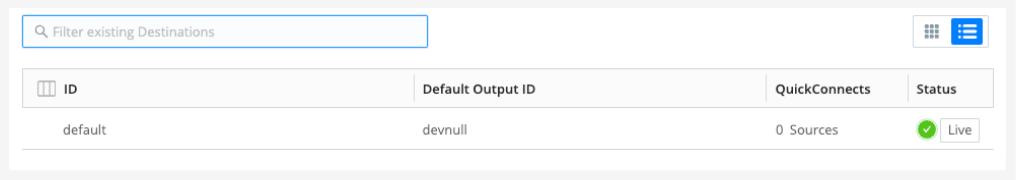
Edit Destination Settings with JSON

To edit any Destination's definition in a JSON text editor, select **Manage as JSON** at the bottom of the Destination modal. You can directly edit multiple values, and you can use the **Import** and **Export** buttons to copy and modify existing Destination configurations as `.json` files.

When JSON configuration contains sensitive information, it is redacted during export.

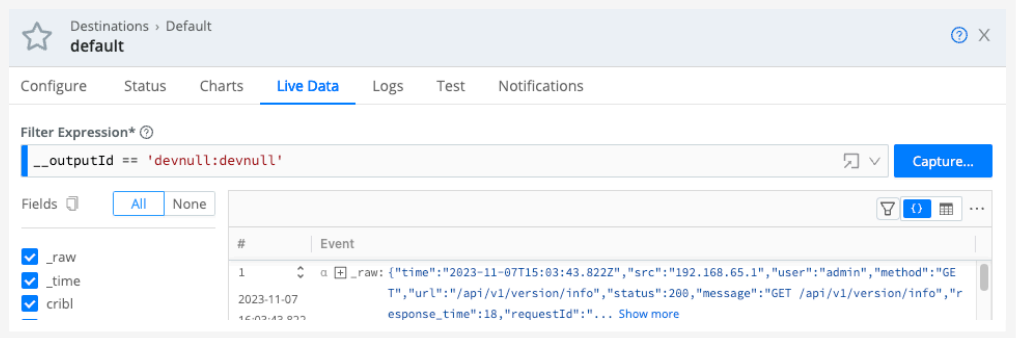
Capture Outgoing Data

To capture data from a single enabled Destination, you can bypass the Data Preview pane and instead capture directly from the list of Destinations. Select the **Live** button beside the Destination you want to capture.



Destination > Live button

You can also start an immediate capture from within an enabled Destination's config modal, by selecting the modal's **Live Data** tab.



Destination modal > Live Data tab

Verify Data Flow

Use the **Test** tab in the Destination's configuration modal to validate the configuration and connectivity of your Destination. A test sends sample events through the Destination to ensure it is correctly set up and can communicate with the downstream receiver.

You can choose from the available sample events or enter your own events, formatted as a JSON array.

Monitor Destination Status

You can get a quick overview of Destination health status by referring to their status icons.

Additionally, each Destination's configuration modal offers two tabs for monitoring: Status and Charts.

Destination Status Icons

Destination status icons are available on the **More** > **Destinations** page and for each individual Destination in the list for a specific Destination type.

The icons have the following meanings:

Icon	Meaning
	Healthy. Operating correctly.
	Warning. Experiencing issues. The Destination is not functioning fully. Specific conditions will depend on the Destination type.
	Critical. Experiencing critical issues. Drill down to the Destination's Status tab to find out the details.
	Disabled. The Destination is configured, but not enabled.
	No health metrics available. This may mean that a Destination is enabled, but has not been deployed yet.
	Inactive. When using GitOps, a Destination appears `Inactive` if its **Environment** field (configured under **Advanced Settings**) does not match the currently active environment determined by the deployed Git branch. This ensures integrations only activate in their designated environments, preventing unintended data flow or misconfiguration.

Destination Status Tab

The **Status** tab provides details about the Edge Nodes in the Fleet and their status. An icon next to each Edge Node uses color to clearly signal its health:

-  : All systems go! Your Edge Node is operating normally.
-  : Attention needed. There's a potential issue with this Edge Node.
-  : Stop! This Edge Node has encountered a critical error.

The way you view Edge Node statuses depends on the size of your Fleet:

- **Fewer than 1000 Edge Nodes**: All Edge Nodes are conveniently displayed on the Status tab, along with their statuses. Use the **Status** checkboxes at the top to filter the list by health (healthy, warning, error). Clicking on any Edge Node row dives deeper, providing specific details to help diagnose issues. The specific set of information provided depends on the Destination type. Keep in mind that this data only reflects process `0` for each Edge Node.
- **More than 1000 Edge Nodes**: With a larger Fleet, you can select a specific Edge Node from the drop-down menu (showing up to 100 Nodes). You can search by `hostname` or `GUID` to find a specific Node. You can also use the **Status** checkboxes to filter which Edge Nodes appear in the drop-down list.



The content of the **Status** tab is loaded live when you open it and only displayed when all the data is ready. With a lot of busy Edge Nodes in a group, or Nodes located far from the Leader, there may be a delay before you see any information.

The statistics presented are reset when the Edge Node restarts.

Destination Health

Cribl Edge evaluates Destination health based on recent delivery outcomes that Edge Nodes report through internal metrics to the Leader.

A Destination changes from **Healthy** when health checks over a recent time window start to show any of the following patterns:

- **Connection and protocol problems:** Repeated connection failures, DNS/TLS errors, or authentication failures between Edge Nodes and the Destination that prevent it from accepting data.
- **HTTP-style delivery failures:** A rising rate of retryable HTTP errors from the downstream service (for example, `5xx` responses, `429` throttling, or request timeouts) as reflected in the Destination's metrics and logs.
- **Backpressure and persistent queue stress:** Backpressure engaging and persistent-queue metrics indicating that the Destination cannot keep up and data is being buffered or blocked.
- **Multi-endpoint imbalance:** For multi-endpoint Destinations such as Splunk Load Balanced or Output Router, a large fraction of senders entering an error state.

There is no fixed hold time for clearing a warning or error state. Once the underlying errors stop and the next health metrics update from the Edge Nodes reflects recovery, the Destination returns to **Healthy** automatically. The exact thresholds and error conditions can vary by destination type, but they always derive from the same internal health metrics.

You can track transitions over time using health metrics (for example, `health.outputs`) in your monitoring system.

Destination Charts Tab

The **Charts** tab presents a visual representation of Destination activity using internal metrics. Unlike the status tab, this data is read almost instantly and does not reset when restarting an Edge Node.

At the top of the modal, a time range picker controls which time window the charts display:

- The picker offers the same presets as the main Monitoring time selector (for example, `5 min`, `15 min`, `1 hour`, `6 hours`, `1 day`).
- When you open the Destination I/O metrics modal from **Monitoring**, the charts inherit the current Monitoring time range.
- When you open the modal from other areas of the UI (for example from **Data** > **Destinations** or directly from a Destination configuration modal), the Charts tab defaults to `10 minutes`, matching the previous behavior.
- Changing the time range in the modal only affects that modal and is not persistent. It does not change the global time range on the originating page (for example, the **Monitoring** overview or the page-level time picker).

Use this picker to zoom in on short spikes or expand to longer windows when you're troubleshooting Destination performance or throughput issues.

The following data is available:

- Total events out
- Average throughput (events per second)
- Total events dropped
- Total bytes out
- Average throughput (bytes per second)
- Backpressure



This data (in contrast with the status tab) is read almost instantly and does not reset when restarting an Edge Node.

12.3.2. Internal

You can send data from Cribl Edge to the following internal receivers:

Destination	Description
Cribl HTTP	Send data between peer Nodes via HTTP
Cribl TCP	Send data between peer Nodes via raw TCP
Cribl Search	Send data to Cribl Search via HTTP
Disk Spool	Spool incoming event data and hold it in a consistent location
Default	Specify a default output from among your Destinations
DevNull	Drop events when testing Pipelines and Routes
Output Router	Routes data to Destinations based on defined rules

12.3.2.1. Cribl HTTP Destination

The Cribl HTTP Destination sends data to a Cribl HTTP Source. Relaying data between a Cribl HTTP Destination and Cribl HTTP Source pair prevents double-billing: You won't incur any additional charges or credit usage after initial ingest to the first Fleet (depending on your Leader setup - see Transfer Data Between Workspaces or Environments for more details).

It's common to send data from Cribl Edge to a Cribl Stream Worker Group using this kind of pairing.



Type: Streaming | TLS Support: Configurable | PQ Support: Yes

For guidance on when to choose this Source versus the Cribl TCP Destination, see Interoperation Protocols.

Within Cribl Stream, the Cribl HTTP Destination is available only in Distributed deployments. In single-instance mode, or for testing, you can use the Webhook Destination instead. However, this substitution will not facilitate sending all internal fields, as described below.

How It Works

You can use the Cribl HTTP Destination to transfer data between Edge Nodes and Workers. If the Cribl HTTP Destination sends data to its Cribl HTTP Source counterpart on another Worker, you're billed for ingress only once - when Cribl first receives the data.

Using a Cribl HTTP Destination/Source pair, you're not charged for subsequent data sent to:

- Workers that share a Leader.
- Workers that don't share a Leader, but are configured in Cribl.Cloud Workspaces within the same Organization.
- Workers that do not share a Leader, but that do have the exact same on-prem license installed.

This use case is common in hybrid Cribl.Cloud deployments, where a customer-managed (hybrid) Worker Node sends data to a Worker in Cribl.Cloud for additional processing and routing to Destinations. However, the Cribl HTTP Destination/Source pair can similarly reduce your metered data ingress in other scenarios, such as on-prem Edge to on-prem Stream.

As one usage example, assume that you want to send data from one Node deployed on-prem, to another that is deployed in Cribl.Cloud. You could do the following:

- Create an on-prem File System Collector (or whatever Collector or Source is suitable) for the data you want to send to Cribl.Cloud.

- Create an on-prem Cribl HTTP Destination.
- Create a Cribl HTTP Source, on the target Stream Worker Group or Edge Fleet in Cribl.Cloud.
- For an on-prem Node configure a File System Collector to send data to the Cribl HTTP Destination, and from there to the Cribl HTTP Source in Cribl.Cloud.
- On Cribl-managed Nodes in Cribl.Cloud, make sure that TLS is either disabled on both the Cribl HTTP Destination and the Cribl HTTP Source it's sending data to, or enabled on both. Otherwise, no data will flow. On Cribl.Cloud instances, the Cribl HTTP Source ships with TLS enabled by default.

Configuration Requirements

The key points about configuring this architecture are:

- This Destination's **Cribl endpoint** field must point to the **Address** and **Port** you've configured on its peer Cribl HTTP Source(s).
- Cribl 3.5.4 was a breakpoint in Cribl HTTP Leader/Worker communications. Nodes running the Cribl HTTP Destination on Cribl 3.5.4 and newer can receive data only from Nodes running v.3.5.4 and newer. Nodes running the Cribl HTTP Destination on Cribl 3.5.3 and older can receive data only from Nodes running v.3.5.3 and older.

Configure a Cribl HTTP Destination

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this Destination definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
    - **Description**: Optionally, enter a description.
    - **Load balancing**: When toggled off (default), if you notice that Cribl Edge is not sending data to all possible IP addresses, enable **Advanced Settings** > **Round-robin DNS**. When toggled on, see Load Balancing Settings below.
    - **Cribl endpoint**: URL of a Cribl Worker to send events to, for example, `http://localhost:10200`.



The **Cribl endpoint** field appears only when you toggle **Load balancing** off. Its value must point to the **Address** and **Port** you've configured on the peer Cribl HTTP Source to which you're sending.

3. Next, you can configure the following **Optional Settings**:
    - **Compression**: Codec to use to compress the data before sending. Defaults to `Gzip`.
    - **Throttling**: Throttle rate per Worker Process, in bytes per second. Defaults to `0` in Cribl Stream, meaning no throttling, and to `256 KB` in Cribl Edge. Multiple-byte units such as KB, MB, GB, and so forth, are also allowed - for example, `42 MB`. When throttling is engaged, your **Backpressure behavior** selection determines whether Cribl Edge will handle excess data by blocking it, dropping it, or queueing it to disk. You can use this setting to limit the risk of overwhelming your deployment when sending high volumes of data from Cribl Edge to Cribl Stream.
    - **Backpressure behavior**: Specifies whether to block, drop, or queue events when all receivers are exerting backpressure. Defaults to `Block`. See Persistent Queue Settings below.
    - **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
4. Optionally, configure any TLS settings, Persistent Queue, Retries, Processing, and Advanced settings outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

Load Balancing Settings

Enabling the **Load balancing** toggle displays the following controls.

Exclude Current Host IPs: Displays under **Optional Setting** when **Load balancing** is toggled on. This toggle determines whether to exclude all IPs of the current host from the list of any resolved hostnames. Defaults to No, which keeps the current host available for load balancing.

Cribl Worker Endpoints: In this table, you specify a set of Cribl Workers on which to load-balance data. To specify more Workers on new rows, click **Add Endpoint**. Each row provides the following fields.

Cribl Endpoint: Enter the URL of a Worker to send events to.



Must point to the **Address** and **Port** configured on a peer Cribl HTTP Source to which you're sending.

Load weight: Set the relative traffic-handling capability for each connection by assigning a weight (> `0`). This column accepts arbitrary values, but for best results, assign weights in the same order of magnitude to all connections. Cribl Edge will attempt to distribute traffic to the connections according to their relative weights.

The final column provides an `X` button to delete any row from the table.

For details on configuring all these options, see About Load Balancing.

TLS Settings (Client Side)

Use TLS: Defaults to toggled off. When toggled on:

Autofill?: This setting is experimental.

Validate server certs: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (for example, the system's CA). Defaults to toggled on. Overrides the **Validate server certs** setting in **Advanced Settings**.

Server name (SNI): Server name for the SNI (Server Name Indication) TLS extension. This must be a host name, not an IP address.

Minimum TLS version: Optionally, select the minimum TLS version to use when connecting.

Maximum TLS version: Optionally, select the maximum TLS version to use when connecting.

Certificate: Select an existing certificate name from the drop-down or add a new certificate by selecting **Create**. See Add a TLS Certificate to a Fleet for details.

CA certificate path: Path on client containing CA certificates (in PEM format) to use to verify the server's cert. Path can reference `$ENV_VARS`.

Private key path (mutual auth): Path on client containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Certificate path (mutual auth): Path on client containing certificates in (PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Passphrase: Passphrase to use to decrypt private key.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)

- Destination Backpressure Triggers

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5` GB. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.



Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Stream Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.

- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Retries

Honor Retry-After header: Toggle on to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream/Edge limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. Any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. Toggle off to ignore all `Retry-After` headers.

Default Retry Behavior: By default, Cribl Stream/Edge applies predefined retry strategies for certain failed requests to the Destination using an exponential backoff algorithm. You can override these settings by manually configuring the retry behavior for a particular error code.



Destinations created before the 4.16.0 release do not support the default `4xx` series retry behavior. If your Destination was created prior to this release and you'd like it to support the behavior, reconfigure the Destination.

To retry requests, Cribl Stream/Edge applies the following rules by default:

Status Code	Action
Any in the `1xx`, `3xx` series	Drop the request
`401`, `403`, `408`, `429` errors	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached
Any other `4xx` series	Drop the request
Any in the `5xx` series	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached

By default, any `429 (Too Many Requests)` response that is not listed in the retry settings is treated as a non-retryable `4xx` error and dropped. If you want `429` responses retried, ensure `429` is included in the **Settings for failed HTTP requests**.

Settings for failed HTTP requests: You can also manually configure Cribl Stream/Edge to automatically retry requests that receive particular HTTP response status codes.

Upon receiving a response code that's on the list, Cribl Stream/Edge first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the

Backoff limit (ms). At that point, Cribl Stream/Edge continues retrying the request without increasing the time between retries any further.

If the sender (which manages the connection to the Destination) is at capacity, it will not accept any incoming events. These incoming events originate internally from a previous stage of the data flow when Destinations send outbound requests to their respective external services, and they include retry requests and new requests. Any events that were already in transit when the sender reached capacity will continue to be processed downstream.

Sender capacity is freed up when an outgoing request succeeds or encounters a non-retryable error. When the sender has available capacity again, it will resume accepting incoming events. This capacity management is influenced by the number of active connections and configured limits, such as concurrency and buffer sizes. If a Pipeline sends events faster than the Destination can process, the buffers may fill up, leading to backpressure and `Sender at capacity` warnings. This backpressure prevents the sender from accepting additional requests until capacity is restored.

For each response code you want to add to the list, select **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: Toggle on to automatically retry requests that have timed out and display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Auth Token

This section is only available for on-prem environments that have a Connected Environment. It is intended for use when migrating data from an on-prem environment to a Cribl.Cloud environment.

If you previously had an on-prem deployment of Cribl and you are moving forward with a Cloud-first or Cloud only solution, you can use connected environments to onboard your data from on-prem Edge Nodes to their counterpart Cribl.Cloud Edge Nodes. See Migrate Data from On-Prem to Cribl.Cloud for implementation details.

Advanced Settings

Validate server certs: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (for example, the system's CA). Defaults to toggled on. This setting is also configurable in **TLS Settings (Client Side)**. If you enable Client-Side TLS, the validation rules defined there will override this setting.

Round-robin DNS: Toggle on to enable round-robin DNS lookup across multiple IP addresses, IPv4 and IPv6. When a DNS server resolves a Fully Qualified Domain Name (FQDN) to multiple IP addresses, Cribl Edge will sequentially use each address in the order they are returned by the DNS server for subsequent connection attempts. Only displayed when the General Settings tab's **Load balancing** option is disabled.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests before blocking. This is set per Worker Process. Defaults to `5`.

Body size limit (KB): Maximum size of the request body before compression. Defaults to `4096` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Body size limit** value until downstream receivers reliably accept all events.

Max events per request: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Extra HTTP headers: Name-value pairs to pass as additional HTTP headers.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Cribl Stream will redact all headers, except non-sensitive headers that you declare below in **Safe headers**.

Safe headers: Add headers to declare them as safe to log in plaintext. (Sensitive headers such as `authorization` will always be redacted, even if listed here.) Use a tab or hard return to separate header names.

Exclude fields: Fields to exclude from the event. This Destination excludes `__kube_*`, `__metadata`, and `__winEvent` by default and forwards all other Internal Fields. For more information about `__winEvent`, see the Windows Event Logs Source topic.



If you are running Cribl Edge 4.0.x (or earlier) and are using the Kubernetes Metrics Source with this Destination, consider excluding the following fields to reduce event size:

```
!__inputId,!__outputId,!__criblMetrics,__* .
```

The contents of `__raw` are often redundant with the `_raw` field's contents. Where they are identical, consider excluding one of the two.

Auth Token TTL minutes: The number of minutes before the internally generated authentication token expires, valid values between 1 and 60.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

The following options are added if you enable the General Settings tab's **Load balancing** option:

DNS resolution period (seconds): Re-resolve any hostnames after each interval of this many seconds, and pick up destinations from A records. Defaults to `600` seconds.

Load balance stats period (seconds): Lookback traffic history period. Defaults to `300` seconds. (Note that If multiple receivers are behind a hostname - i.e., multiple A records - all resolved IPs will inherit the weight of the host, unless each IP is specified separately. In Cribl Stream load balancing, IP settings take priority over those from hostnames.)

Internal Fields Loopback to Sources

The Cribl HTTP and Cribl TCP Destinations differ from all other Destinations in the way they handle internal fields: They normally send data back to their respective Cribl Sources - where Cribl internal fields, metrics, and sender-generated fields can all be useful.

These Destinations forward all internal fields by default, except for any that you exclude in **Advanced Settings** > **Exclude fields**.

As examples, if the following fields are present on an event forwarded by a Cribl HTTP or Cribl TCP Destination, they'll be accessible in the ingesting Cribl HTTP/TCP Source: `__criblMetrics`, `__srcIpPort`, `__inputId`, and `__outputId`.

Proxying Requests

If you need to proxy HTTP/S requests, see System Proxy Configuration.

12.3.2.2. Cribl TCP Destination

The Cribl TCP Destination sends data to a Cribl TCP Source. Relaying data between a Cribl TCP Destination and Cribl TCP Source pair prevents double-billing: You won't incur any additional charges or credit usage after initial ingest to the first Fleet (depending on your Leader setup - see Transfer Data Between Workspaces or Environments for more details).

It's common to send data from Cribl Edge to a Cribl Stream Worker Group using this kind of pairing.



Type: Streaming | TLS Support: Configurable | PQ Support: Yes

For guidance on when to choose this Source versus the Cribl HTTP Destination, see Interoperation Protocols.

Within Cribl Stream, the Cribl TCP Destination is available only in Distributed deployments. In single-instance mode, or for testing, you can use the TCP JSON Destination instead. However, this substitution will not facilitate sending all internal fields, as described below.

How It Works

You can use the Cribl TCP Destination to transfer data between Edge Nodes and Workers. If the Cribl TCP Destination sends data to its Cribl TCP Source counterpart, you're billed for ingress only once - when Cribl first receives the data.

Using a Cribl HTTP Destination/Source pair, you're not charged for subsequent data sent to:

- Workers that share a Leader.
- Workers that don't share a Leader, but are configured in Cribl.Cloud Workspaces within the same Organization.
- Workers that do not share a Leader, but that do have the exact same on-prem license installed.

This use case is common in hybrid Cribl.Cloud deployments, where a customer-managed (hybrid) Worker Node sends data to a Worker in Cribl.Cloud for additional processing and routing to Destinations. However, the Cribl TCP Destination/Source pair can similarly reduce your metered data ingress in other scenarios, such as on-prem Cribl Edge to on-prem Cribl Stream.

As one usage example, assume that you want to send data from one Node deployed on-prem, to another that is deployed in Cribl.Cloud. You could do the following:

- Create an on-prem File System Collector (or whatever Collector or Source is suitable) for the data you want to send to Cribl.Cloud.

- Create an on-prem Cribl TCP Destination.
- Create a Cribl TCP Source, on the target Cribl Stream Worker Group or Cribl Edge Fleet in Cribl.Cloud.
- For an on-prem Node configure a File System Collector to send data to the Cribl TCP Destination, and from there to the Cribl TCP Source in Cribl.Cloud.
- On Cribl-managed Nodes in Cribl.Cloud, make sure that TLS is either disabled on both the Cribl TCP Destination and the Cribl TCP Source it's sending data to, or enabled on both. Otherwise, no data will flow. On Cribl.Cloud instances, the Cribl TCP Source ships with TLS enabled by default.

Configuration Requirements

The key points about configuring this architecture are:

- When you configure the Cribl TCP Destination, its **Address** and **Port** values must point to the **Address** and **Port** you've configured on the Cribl TCP Source.
- Cribl 3.5.4 was a breakpoint in Cribl TCP Leader/Worker communications. Nodes running the Cribl TCP Source on Cribl 3.5.4 and newer can send data only to Nodes running v.3.5.4 and newer. Nodes running the Cribl TCP Source on Cribl 3.5.3 and older can send data only to Nodes running v.3.5.3 and older.

Configure a Cribl TCP Destination

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this Destination definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
    - **Description**: Optionally, enter a description.
    - **Load balancing**: When toggled on, see Load Balancing Settings below.



The following two fields appear **only** if **Load balancing** is toggled off (default), and must match the **Address** and **Port** you've configured on the peer Cribl TCP Source to which you're sending.

- **Address**: Hostname of the receiver.
- **Port**: Port number to connect to on the host, for example, `10300`.

3. Next, you can configure the following **Optional Settings**:
   - **Compression**: Codec to use to compress the data before sending. Defaults to `Gzip`.
   - **Throttling**: Throttle rate, in bytes per second. Defaults to `0`, meaning no throttling. Multiple-byte units such as KB, MB, and so forth, are also allowed - for example, `42 MB`. When throttling is engaged, your **Backpressure behavior** selection determines whether Cribl Edge will handle excess data by blocking it, dropping it, or queueing it to disk. When used in conjunction with **Connection limit**, the throttling rate is applied per connection. For instance, if **Connection limit** is set to `2` and throttling is set to `3 MB`, each of the two connections can send data at a rate of up to 3 MB per second, resulting in a total potential throughput of 6 MB per second across both connections.
   - **Backpressure behavior**: Specifies whether to block, drop, or queue events when all receivers are exerting backpressure. Defaults to `Block`. See Persistent Queue Settings below.
   - **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
4. Optionally, configure any TLS settings, Persistent Queue, Timeout, Processing, and Advanced settings outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

Load Balancing Settings

Enabling the **Load balancing** toggle displays the following controls:

Exclude Current Host IPs

Displays under **Optional Setting** when **Load balancing** is toggled on. This toggle determines whether to exclude all IPs of the current host from the list of any resolved hostnames. Default is toggled off, which keeps the current host available for load balancing.

Destinations

The **Destinations** table is where you specify a set of receivers on which to load-balance data. Click **Add Destination** to specify more receivers on new rows. Each row provides the following fields:

Address: Hostname of a receiver. Optionally, you can paste in a comma-separated list, in `<host>:<port>` format.

Port: Port number to send data to on the host.

Each **Address**/**Port** combination must match the **Address** and **Port** configured on a peer TCP Source to which you're sending.

TLS: Whether to inherit TLS configs from group setting, or disable TLS. Defaults to `inherit`.

TLS servername: Servername to use if establishing a TLS connection. If not specified, defaults to connection host (if not an IP); otherwise, uses the global TLS settings.

Load weight: Set the relative traffic-handling capability for each connection by assigning a weight (> `0`). This column accepts arbitrary values, but for best results, assign weights in the same order of magnitude to all connections. Cribl Edge will attempt to distribute traffic to the connections according to their relative weights.

The final column provides an `X` button to delete any row from the table.

For details on configuring all these options, see About Load Balancing.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.



If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5 GB`. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10

seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.



Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

TLS Settings (Client Side)

Use TLS: Defaults to toggled off. When toggled on:

Autofill?: This setting is experimental.

Validate server certs: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (for example, the system's CA). Default is toggled off.

Server name (SNI): Server name for the SNI (Server Name Indication) TLS extension. This must be a host name, not an IP address.

Minimum TLS version: Optionally, select the minimum TLS version to use when connecting.

Maximum TLS version: Optionally, select the maximum TLS version to use when connecting.

Certificate name: The name of the predefined certificate.

CA certificate path: Path on client containing CA certificates (in PEM format) to use to verify the server's cert. Path can reference `$ENV_VARS`.

Private key path (mutual auth): Path on client containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Certificate path (mutual auth): Path on client containing certificates in (PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Passphrase: Passphrase to use to decrypt private key.

Timeout Settings

Connection timeout: Amount of time (in milliseconds) to wait for the connection to establish before retrying. Defaults to `10000`.

Write timeout: Amount of time (in milliseconds) to wait for a write to complete before assuming connection is dead. Defaults to `60000`.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Stream Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Auth Token

This section is only available for on-prem environments that have a Connected Environment. It is intended for use when migrating data from an on-prem environment to a Cribl.Cloud environment.

If you previously had an on-prem deployment of Cribl and you are moving forward with a Cloud-first or Cloud only solution, you can use connected environments to onboard your data from on-prem Edge Nodes to their counterpart Cribl.Cloud Edge Nodes. See Migrate Data from On-Prem to Cribl.Cloud for implementation details.

Advanced Settings

Auth Token TTL minutes: The number of minutes before the internally generated authentication token expires, valid values between 1 and 60.

Exclude fields: Fields to exclude from the event. This Destination excludes `__kube_*`, `__metadata`, and `__winEvent` by default and forwards all other Internal Fields. For more information about `__winEvent`, see the Windows Event Logs Source topic.

If you are running Cribl Edge 4.0.x (or earlier) and are using the Kubernetes Metrics Source with this Destination, consider excluding the following fields to reduce event size:

`!__inputId,!__outputId,!__criblMetrics,__* .`

The contents of `__raw` are often redundant with the `_raw` field's contents. Where they are identical, consider excluding one of the two.

Log failed requests to disk: Toggling on makes the payload of the most recently failed request available for inspection. See Inspect Payload to Troubleshoot Closed Connections below.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

The following options are added if you enable the General Settings tab's **Load balancing** option:

DNS resolution period (seconds): Re-resolve any hostnames after each interval of this many seconds, and pick up destinations from A records. Defaults to `600` seconds.

Load balance stats period (seconds): Lookback traffic history period. Defaults to `300` seconds. (Note that If multiple receivers are behind a hostname - i.e., multiple A records - all resolved IPs will inherit the weight of the host, unless each IP is specified separately. In Cribl Stream load balancing, IP settings take priority over those from hostnames.)

Max connections: Constrains the number of concurrent receiver connections, per Worker Process, to limit memory utilization. If set to a number > `0`, then on every DNS resolution period, Cribl Edge will randomly select this subset of discovered IPs to connect to. Cribl Edge will rotate IPs in future resolution periods - monitoring weight and historical data, to ensure fair load balancing of events among IPs.

Inspect Payload to Troubleshoot Closed Connections

When a downstream receiver closes connections from this Destination (or just stops responding), inspecting the payload of the most recently failed request can help you find the cause. For example:

- Suppose you send an event whose size is larger than the downstream receiver can handle.
- Suppose you send an event that has a `number` field, but the value exceeds the highest number that the downstream receiver can handle.

When **Log failed requests to disk** is enabled, you can inspect the last failed request payload. Here is how:

1. In the Destination UI, navigate to the **Logs** tab.
2. Find a log entry with a `connection error` message.
3. Expand the log entry.

4. If the message includes the phrase `See payload file for more info`, note the path in the `file` field on the next line.

Now you have the path to the directory where Cribl Edge is storing the payload from the last failed request.

Internal Fields Loopback to Sources

The Cribl TCP and Cribl HTTP Destinations differ from all other Destinations in the way they handle internal fields: They normally send data back to their respective Cribl Sources - where Cribl internal fields, metrics, and sender-generated fields can all be useful.

These Destinations forward all internal fields by default, except for any that you exclude in **Advanced Settings** > **Exclude fields**.

As examples, if the following fields are present on an event forwarded by a Cribl HTTP or Cribl TCP Destination, they'll be accessible in the ingesting Cribl HTTP/TCP Source: `__criblMetrics`, `__srcIpPort`, `__inputId`, and `__outputId`.

12.3.2.3. Cribl Search Destination

The Cribl Search Destination enables Cribl Edge to send events directly to Cribl Search over a dedicated, internal HTTP path.

Use the Cribl Search Destination when you want to send data from:

- Fleets to Cribl Search running in the same Cribl.Cloud organization.
- Fleets to a connected Cribl.Cloud environment that hosts Cribl Search.
- Fleets to Cribl Search as an ingestion endpoint, instead of configuring generic Cribl HTTP Destinations by hand.



Type: Streaming | TLS Support: Yes | PQ Support: Yes

For Cribl Stream Worker Groups that are managed in Cribl.Cloud and sending to Cribl Search in the same Organization, TLS for Cribl-to-Cribl traffic is managed automatically. You do not configure TLS settings directly on this Destination.

How It Works

With the Cribl Search Destination:

- The Destination serializes events using the same HTTP payload format as Cribl HTTP.
- Events are delivered over HTTP to a Cribl HTTP Source in the Cribl.Cloud environment that hosts Cribl Search. That Source is preconfigured to forward all received events into Cribl Search's ingestion pipeline.
    - That Cribl HTTP Source typically runs in a different Fleet than the one where you configured the Cribl Search Destination, and it serves as the HTTP entry point for Cribl Search ingestion.
- Cribl Search indexes the events according to its own dataset and schema configuration.

Data Transfer Behavior

You can use the Cribl Search Destination to transfer data between Fleets and Cribl Search without incurring additional ingest charges, as long as it sends data to a Cribl HTTP Source that acts as the entry point for Cribl Search ingestion, and your deployment meets the eligibility rules in Transfer Data Between Workspaces or Environments.

If the Cribl Search Destination sends data to its Cribl HTTP Source counterpart for Search on another Edge Node, you're billed for ingress only once when Cribl first receives the data. Using a Cribl Search Destination and

Cribl HTTP Source pair, you're not charged for subsequent data sent to:

- Edge Nodes that share a Leader.
- Edge Nodes that don't share a Leader but are configured in Cribl.Cloud Workspaces within the same Organization.
- Edge Nodes that do not share a Leader but that have the exact same on-prem license installed.

Prerequisites

Before configuring the Cribl Search Destination:

- You must have Cribl Search enabled in a Cribl.Cloud organization, with at least one Lakehouse Engine configured to ingest data from the Cribl HTTP Source used by this Destination.
- For cross-environment or cross-Workspace paths (for example, on-prem Stream to Cloud Search), both environments must be on paid plans and eligible for free data transfer via Cribl TCP/HTTP.
    - For those paths, configure Cribl-to-Cribl authentication using Auth Tokens on both the Cribl HTTP Source in Cribl.Cloud that receives events for Cribl Search, and on the on-prem Cribl Search Destination as described in Transfer Data Between Workspaces or Environments.

For Worker Groups running in Cribl.Cloud and sending to Cribl Search in the same Organization, the Cribl Search Destination uses the existing Cribl-to-Cribl authentication path automatically. No additional tokens are required.

Configuration Requirements

The Cribl Search Destination's **Cribl endpoint** must point to the **Address** and **Port** configured on the corresponding Cribl HTTP Source that serves as the HTTP entry point for Cribl Search in the target Fleet.

Configure a Cribl Search Destination

1. On the top bar, select **Products**, and then select Cribl Edge. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination modal**, configure the following under **General Settings**:

- **Output ID**: Enter a unique name to identify this Destination definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original Output ID.
- **Description**: Optionally, enter a description.
- **Cribl endpoint**: URL of the Cribl HTTP Source that receives events for Cribl Search. In Cribl.Cloud this is auto-populated as `https://search.<workspace>.<organizationId>.cribl.cloud:10200`. In self-managed or connected environments, use the host and port from your Cribl HTTP Source.

3. Next, you can configure the following **Optional Settings**:
   - **Throttling**: Throttle rate per Worker Process, in bytes per second. Defaults to `0`, meaning no throttling. Multiple-byte units such as KB, MB, GB, and so forth, are also allowed (for example, 42 MB). When throttling is engaged, your backpressure behavior selection determines whether Cribl Edge will handle excess data by blocking it, dropping it, or queueing it to disk.
   - **Backpressure behavior**: Specifies whether to block, drop, or queue events when all receivers are exerting backpressure. Defaults to **Block**. See Persistent Queue Settings below.
   - **Tags**: Optionally, add tags that you can use to filter and group Destinations on the Destinations page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
4. Optionally, configure any Processing Settings, Auth Tokens, and Advanced Settings settings outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

> This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

> If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5 GB`. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.



Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Retries

Honor Retry-After header: Toggle on to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream/Edge limits the delay to 180 seconds even if the `Retry-After`

header specifies a longer delay. Any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. Toggle off to ignore all `Retry-After` headers.

Default Retry Behavior: By default, Cribl Stream/Edge applies predefined retry strategies for certain failed requests to the Destination using an exponential backoff algorithm. You can override these settings by manually configuring the retry behavior for a particular error code.



Destinations created before the 4.16.0 release do not support the default `4xx` series retry behavior. If your Destination was created prior to this release and you'd like it to support the behavior, reconfigure the Destination.

To retry requests, Cribl Stream/Edge applies the following rules by default:

Status Code	Action
Any in the `1xx`, `3xx` series	Drop the request
`401`, `403`, `408`, `429` errors	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached
Any other `4xx` series	Drop the request
Any in the `5xx` series	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached

By default, any `429 (Too Many Requests)` response that is not listed in the retry settings is treated as a non-retryable `4xx` error and dropped. If you want `429` responses retried, ensure `429` is included in the **Settings for failed HTTP requests**.

Settings for failed HTTP requests: You can also manually configure Cribl Stream/Edge to automatically retry requests that receive particular HTTP response status codes.

Upon receiving a response code that's on the list, Cribl Stream/Edge first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream/Edge continues retrying the request without increasing the time between retries any further.

If the sender (which manages the connection to the Destination) is at capacity, it will not accept any incoming events. These incoming events originate internally from a previous stage of the data flow when Destinations send outbound requests to their respective external services, and they include retry requests and new requests. Any events that were already in transit when the sender reached capacity will continue to be processed downstream.

Sender capacity is freed up when an outgoing request succeeds or encounters a non-retryable error. When the sender has available capacity again, it will resume accepting incoming events. This capacity management is influenced by the number of active connections and configured limits, such as concurrency and buffer sizes. If a Pipeline sends events faster than the Destination can process, the buffers may fill up, leading to backpressure and `Sender at capacity` warnings. This backpressure prevents the sender from accepting additional requests until capacity is restored.

For each response code you want to add to the list, select **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: Toggle on to automatically retry requests that have timed out and display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Auth Tokens

This section is only available for on-prem environments that have a Connected Environment. It is intended for use when migrating data from an on-prem environment to a Cribl.Cloud environment.

If you previously had an on-prem deployment of Cribl and you are moving forward with a Cloud-first or Cloud only solution, you can use connected environments to onboard your data from on-prem Worker Nodes to their counterpart Cribl.Cloud Worker Nodes. See Migrate Data from On-Prem to Cribl.Cloud for implementation details.

Advanced Settings

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests before blocking. This is set per Worker Process. Defaults to `5`.

Body size limit (KB): Maximum size of the request body before compression. Defaults to `4096` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Body size limit** value until downstream receivers reliably accept all events.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Cribl Stream will redact all headers, except non-sensitive headers that you declare below in **Safe headers**.

Safe headers: Add headers to declare them as safe to log in plaintext. (Sensitive headers such as `authorization` will always be redacted, even if listed here.) Use a tab or hard return to separate header names.

Internal Fields Loopback to Sources

Because the Cribl Search Destination is implemented as a Cribl HTTP Destination, it shares the same internal-fields behavior as the Cribl HTTP and Cribl TCP Destinations. These Destinations normally send data back to their respective Cribl Sources, where Cribl internal fields, metrics, and sender-generated fields can be useful. They forward all internal fields by default, except for any that you exclude in **Advanced Settings** > **Exclude fields**. For example, if `__criblMetrics`, `__srcIpPort`, `__inputId`, or `__outputId` are present on an event forwarded by the Cribl Search Destination, they'll be accessible in the ingesting Cribl HTTP Source.

Proxying Requests

If you need to proxy HTTP/S requests, see System Proxy Configuration.

12.3.2.4. Disk Spool Destination

The **Disk Spool** Destination writes recent events to disk, so you can spool incoming event data and save it in a predefined file path location. This consistent path can help you find data when you use generic Cribl Search Datasets.

Type: **Internal** | TLS Support: **N/A** | PQ Support: **No**

Configure Cribl Edge to Spool Event Data

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this Disk Spool Destination definition.
    - **Description**: Optionally, enter a description.
3. Next, you can configure the following **Optional Settings**:
    - **Bucket time span**: The amount of time that data is held in each bucket before it's written to disk. The default is 10 minutes (`10m`). Consider the volume of the data source, as high-volume logs can fill a disk up quickly if you enter a long time span.
    - **Data size limit**: This is the maximum amount of disk space that spooled data is allowed to consume. Once data reaches this amount, Edge will delete data starting with the oldest buckets.
    - **Data age limit**: The duration of time for which Cribl Edge will retain data. Edge will delete data that exceeds the max age entered in this field. Example values are `2h`, `4d`. The default value is 24 hours (`24h`).
    - **Compression**: The codec that Edge uses to compress the data. The default is `gzip`.
    - **Partitioning expression**: The JavaScript expression that defines how Edge partitions and organizes files into time bucket directories. The default is date-based. If this field is left blank, Cribl Edge will fall back to the event's `__partition` field value (if present), or otherwise to the root directory of the Output Location and Staging Location.

- **Tags**: Add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

4. Optionally, configure any Processing and Advanced settings outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

> **Use Case Example**: When you partition incoming events, Search can find them more efficiently. Imagine you've configured three Sources, each sending event data to the Disk Spool Destination. Enter the partitioning expression `__inputId`. Now you can send data from the three different Sources to individual subdirectories under the spool's time bucket directories.

For help with writing partitioning expressions, check out Improving Partitioning Expressions.

Processing Settings Post-Processing

Pipeline: The Pipeline that will process the data before sending it out.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Advanced Settings

Environment: If you're using GitOps, use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Collect Event Data

Once your Destination is configured, you can begin sending data to it from any Source. We recommend only connecting Sources that do not support disk spooling at the Source level. When you have disk spooling enabled at the Source *and* you connect the Source to the Disk Spool Destination, event data is duplicated and sent into two different spools.

When you have the Disk Spool Destination configured and Sources connected to it, check that data is flowing using the **Live Data** tab in the Disk Spool Destination.

Next, use Cribl Search to locate spooled data.

Searching for Spooled Data

Disk spooling allows you to store recent events to disk, and makes them available to find with Cribl Search. Typically, data from Edge is sent to and stored directly in a Destination. By caching data in a spool, you can perform data queries in real time without needing to connect to the data's final destination.

For example, let's say you're an IT administrator who is troubleshooting issues with a specific Edge Node. Rather than navigating away from Edge to access the software that's hosting all the Edge data and looking through logs, you can stay in Cribl Edge and run a Search query, returning data for a specific Node over a duration of time you define.

To find spooled data, make sure you have at least one Source connected to the Disk Spool Destination, then open Cribl Search.

1. In the search query box, type `dataset="cribl_edge_spool"` to return all events currently in the spool.
2. Use statements, scope, and processing operators to refine your spooled data. Check out Build a Search for more information and examples.

12.3.2.5. Default Destination

The **Default** Destination simply enables you to specify a default output from among your already-configured Destinations.



Type: Internal | TLS Support: N/A | PQ Support: N/A

Configure Cribl Edge's Default Destination

In Cribl Edge configure Default.

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). A **Default** Destination is preconfigured and ready to use in the right **Destinations** column. Hover over the tile and click its **Configure** button to proceed.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. From the resulting **Destinations** page, click anywhere on the `default` row to proceed.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: This is prefilled with the default value `default`, which cannot be changed via the UI.
    - **Default Output ID** drop-down to select one of your configured Destinations.
3. Select **Save**, then **Commit & Deploy**.

Preventing Circular References

If you've configured an Output Router Destination with a branch that points to this Default Destination (`default:default`), you cannot select that Output Router here. This restriction prevents a circular dependency.

12.3.2.6. DevNull Destination

The DevNull Destination simply drops events. Cribl provides this as a basic output to test Pipelines and Routes.



Type: Internal | TLS Support: N/A | PQ Support: N/A

Configure Cribl Edge to Forward to DevNull

DevNull requires no configuration: A DevNull Destination is preconfigured and active as soon as you install Cribl Edge. To verify this, follow the direction below.

On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:

- To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). A **DevNull** Destination is preconfigured and ready to use in the right **Destinations** column.
- To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. From the resulting **Destinations** page, click anywhere on the `devnull` row to proceed.

12.3.2.7. Output Router Destination

The Output Router Destination enables dynamic, rule-based routing of events to one or more downstream Destinations. It acts as a meta-destination, evaluating each event against an ordered set of filter rules (JavaScript expressions) to determine the appropriate Destinations (outputs).

This is useful where routing decisions depend on event content, such as sending security logs to a SIEM and application logs to cloud storage, all from a single Pipeline.

How It Fits Into the Data Flow

- **Source ingestion**: Events are ingested from Sources.
- **Pre-processing Pipelines**: Pre-processing Pipelines attached to Sources run first. Fields created or modified here are available for routing and Output Router rule evaluation.
- **Route Pipelines**: If configured, regular Pipelines attached to Routes process events next. Fields set here are also available to Output Router rules.
- **Output Router**: The Output Router evaluates its rules (**Filter expressions**) against each event using fields from pre-processing and Route Pipelines. Matching rules determine which Destinations receive the event. If multiple rules match and final is `false`, the event is cloned and sent to all matching Destinations.

  Events that do not match a rule are sent to the Default Destination.
- **Destination(s)**: Events are delivered to the selected Destinations.
- **Post-processing Pipelines**: Post-processing Pipelines attached to Destinations run last, after routing. Fields created or modified here are not available to Output Router rules.

Backpressure and Persistent Queue Behavior

The Output Router is a meta-destination that evaluates rules and forwards events to downstream Destinations. It does not provide its own persistent queue. Backpressure behavior is controlled by each downstream Destination.

If multiple rules match (with **Final** toggled off), Cribl Edge clones the event and sends it to each matching Destination. If one of those Destinations is configured to **Block**, it can block delivery for that event across all Destinations. If one is configured for **Persistent Queue**, Cribl Edge can buffer that copy of the event locally while other healthy Destinations continue to receive data, until the queue fills or the downstream Destination recovers.

For more information, see About Persistent Queues.

Type: Internal | TLS Support: N/A | PQ Support: N/A

Configure an Output Router Destination

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Router name**: Enter a unique name to identify this Router definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
    - **Description**: Optionally, enter a description.
    - **Rules**: A list of event routing rules. Rules are evaluated in order, top-down. For details, see Rules.
3. You can configure the following **Optional Settings**:
    - **System fields**: A list of fields to automatically add to events that use this Destination. For details, see System Fields.
    - **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
4. On the **Advanced Settings** tab you can configure an **Environment** if you're using GitOps. Optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.
5. Select **Save**, then **Commit & Deploy**.

Rules

Rules route events by examining event content. These rules are processed sequentially from top to bottom.

Filter expression: A JavaScript expression to select events to route to Destinations. Filter expressions evaluate event fields after pre-processing (input conditioning at the Source) and after any Route Pipeline processing, but before any post-processing (output conditioning at the Destination). Fields created or modified in pre-

processing or Route Pipelines are available to Output Router filter expressions. Fields created or modified during post-processing are not available, as post-processing occurs after routing decisions are made.



Use the expression `true` to match any remaining events you want sent, like in this example.

Output: The Destination to send matching events to. This can be any configured Destination, except for another Output Router or a Default Destination that points back to an Output Router.

Description: Optionally, enter a description of this rule's purpose.

Final: By default (toggled on), event routing stops after this rule matches. Toggle off if you want the event to be evaluated against other rules lower in the stack, allowing cloning to multiple Destinations.

System Fields

By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Example

Scenario:

- Send all events where `host` starts with `66` to Destination `San Francisco`.
- From the rest of the events: Send all events with `method` field `POST` or `GET` to both `Seattle` and `Los Angeles` (i.e., clone them).
- Send the remaining events to `New York City`.

Router Name: **router66**

Filter Expression	Output	Final
`host.startsWith('66')`	**San Francisco**	Yes
`method=='POST' || method=='GET`	**Seattle**	No

Filter Expression	Output	Final
`method=='POST' || method=='GET'`	**Los Angeles**	Yes
`true`	**New York**	Yes

12.3.3. Amazon

You can send data from Cribl Edge to the following Amazon services:

Destination	Description
Amazon CloudWatch Logs	Send data to Amazon CloudWatch Logs
Amazon Kinesis Data Streams	Deliver data to an Amazon Kinesis Data Stream
Amazon S3 Compatible Stores	Send data to Amazon S3 or to an S3-compatible store
Amazon SQS	Send events to Amazon Simple Queuing Service
Amazon Security Lake	Send OCSF data to Amazon Security Lake in Parquet format

12.3.3.1. Amazon CloudWatch Logs Destination

Cribl Edge supports sending data to Amazon CloudWatch Logs. Cribl Edge does **not** have to run on AWS in order to deliver data to CloudWatch Logs.



Type: Streaming | TLS Support: Yes | PQ Support: Yes

IAM Permissions

Ensure your AWS IAM user or role has the following permissions:

- `logs:CreateLogGroup` - Allows the creation of new log groups in CloudWatch Logs.
- `logs:CreateLogStream` - Permits the creation of new log streams within a log group.
- `logs:PutLogEvents` - Enables the insertion of log events into a log stream.

Configure Cribl Edge to Output to Amazon CloudWatch Logs

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this CloudWatch definition.
    - **Description**: Optionally, enter a description.
    - **Log group name**: CloudWatch log group to associate events with.
    - **Log stream prefix**: Prefix for CloudWatch log stream name. This prefix will be used to generate a unique log stream name per Cribl Edge instance. (For example, `myStream_myHost_myOutputId`.)
    - **Region**: AWS Region where the CloudWatch Logs group is located.
3. Next, you can configure the following **Optional Settings**:

- **Backpressure behavior**: Select whether to block, drop, or queue events when all receivers are exerting backpressure. (Causes might include a broken or denied connection, or a rate limiter.) Defaults to `Block`. When this field is set to `queue`, see Persistent Queue Settings for details.
- **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

4. Optionally, configure any Authentication, Assume Role, Processing, and Advanced settings outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

> On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.
>
> This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.

Mode	Description
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.



If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5` GB. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.



Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Authentication

Use the **Authentication method** drop-down to select an AWS authentication method.

Auto: This default option uses the AWS SDK for JavaScript to automatically obtain credentials in the following order of attempts:

1. **Environment Variables**: Loaded from environment variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`.
2. **IAM Identity Center (SSO)**: Loaded from AWS IAM Identity Center configuration.
3. **Shared Credentials File**: Loaded from the shared credentials file (`~/.aws/credentials`).
4. **IAM Roles for Amazon EC2/ECS**: Loaded from AWS Identity and Access Management (IAM) roles attached to an EC2 instance or ECS container.
5. **JSON File on Disk**: Loaded from a JSON file on disk.
6. **Other Credential-Provider Classes**: Other credential-provider classes provided by the AWS SDK for JavaScript.

The `Auto` method works both when running on AWS and in other environments where the necessary credentials are available through one of the above methods.



SSO Providers

When using the auto authentication method, you can leverage SSO providers like SAML and Okta to issue temporary credentials. These credentials should be set in the environment variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`. The AWS SDK will then use these environment variables to authenticate.

Manual: If not running on AWS, you can select this option to enter a static set of user-associated IAM credentials (your access key and secret key) directly or by reference. This is useful for Workers not in an AWS VPC, for example, those running a private cloud. The **Manual** option exposes these corresponding additional fields:

- **Access key**: Enter your AWS access key. If not present, will fall back to the `env.AWS_ACCESS_KEY_ID` environment variable, or to the metadata endpoint for IAM role credentials.
- **Secret key**: Enter your AWS secret key. If not present, will fall back to the `env.AWS_SECRET_ACCESS_KEY` environment variable, or to the metadata endpoint for IAM credentials.

Secret: If not running on AWS, you can select this option to supply a stored secret that references an AWS access key and secret key. The **Secret** option exposes this additional field:

- **Secret key pair**: Use the drop-down to select an API key/secret key pair that you've configured in Cribl Edge's secrets manager. A **Create** link is available to store a new, reusable secret.

Assume Role

When using **Assume Role** to access resources in a different Region than Cribl Edge, you can target the AWS Security Token Service (STS) endpoint specific to that Region by using the `CRIBL_AWS_STS_REGION` environment variable on your Edge Node. To specify a custom STS endpoint, set the `CRIBL_AWS_STS_ENDPOINT` environment variable. If both are set, `CRIBL_AWS_STS_ENDPOINT` takes precedence. Setting an invalid Region or endpoint results in a fallback to the global STS endpoint.

Enable for CloudWatch Logs: Toggle on to use Assume Role credentials to access CloudWatch Logs.

AssumeRole ARN: Enter the Amazon Resource Name (ARN) of the role to assume.

External ID: Enter the External ID to use when assuming role.

Duration (seconds): Duration of the Assumed Role's session, in seconds. Minimum is 900 (15 minutes). Maximum is 43200 (12 hours). Defaults to 3600 (1 hour).

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.

- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Advanced Settings

Endpoint: CloudWatch Logs service endpoint. If empty, defaults to AWS' Region-specific endpoint. Otherwise, use this field to point to a CloudWatchLogs-compatible endpoint.



To proxy outbound HTTP/S requests, see System Proxy Configuration.

Queue size limit: Maximum number of queued batches before blocking. Defaults to `5`.

Record size limit (KB, uncompressed): Maximum size of each individual record before compression. For non-compressible data, 1MB (the default) is the maximum recommended size.

Flush period (sec): Maximum time between requests. Low settings could cause the payload size to be smaller than its configured maximum.

Reuse connections: Whether to reuse connections between requests. Toggling on (default) can improve performance.

Reject unauthorized certificates: Whether to accept certificates that cannot be verified against a valid Certificate Authority (for example, self-signed certificates). Defaults to toggled on.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

How the Amazon CloudWatch Logs Destination Batches Events

Amazon CloudWatch Logs specifies that a batch of log events in a single request cannot span more than 24 hours.

When the Amazon CloudWatch Logs Destination transmits a batch of events, it notes the most recent event in the batch (by date-time) and drops all events in the batch that are 24 hours older (or more) than the most recent event.

To monitor dropped events, check **Out of Range Dropped Event Count** under **Status**. Alternately, you can monitor a debug log action to flag how many events are being dropped in a batch.

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

12.3.3.2. Amazon Kinesis Data Streams Destination

Cribl Edge can output events to **Amazon Kinesis Data Streams** records of up to 1MB uncompressed. Cribl Edge does **not** have to run on AWS in order to deliver data to a Kinesis Data Stream.



Type: Streaming | TLS Support: Yes | PQ Support: Yes

Meeting the Needs of Different Downstream Receivers

You can use this Destination to send data to different kinds of downstream receivers that have different expectations. In all cases what the Destination sends out are Kinesis Records. This works because the Amazon Kinesis Data Streams Service is flexible about what is actually in a Record, as the two following use cases show.

Sending Multiple Events within One Record

When used to feed data into Amazon Data Firehose, this Destination will start with multiple events as NDJSON, gzip-compress them together into a blob, and send that as a single Kinesis Record. This uses the Kinesis PutRecord API call to send the combined events in a single Record.

Amazon Data Firehose will gunzip-uncompress the Record and split the NDJSON out into individual events again. There are also other downstream receivers besides Amazon Data Firehose that can do the same thing.

This approach requires either:

- **Advanced Settings** > **Compression** set to `Gzip`, **or**
- **Advanced Settings** > **Send batched** toggled on.

Meanwhile, if you want Cribl Edge to be the consumer, on the Kinesis Streams Source, set **Record data format** to `Cribl`. See the format details below.

Sending One Event Per Record

In contrast to the first use case, some downstream receivers require a Kinesis Record to contain exactly one Cribl event. Here, this Destination uses the Kinesis PutRecords API call to send multiple events in a single API call, but where each event has a Record to itself. (Note the difference between `PutRecord` in the previous use case and `PutRecords` in this one.)

This approach requires:

- **Advanced Settings** > **Compression** set to `None`, **and**
- **Advanced Settings** > **Send batched** toggled off.

Meanwhile, if you want Cribl Edge to be the consumer, on the Kinesis Streams Source, set **Record data format** to `Newline JSON`. See the format details below.

Amazon Kinesis Data Streams Permissions

The following permissions are needed to write to an Amazon Kinesis Stream:

Shard Verification (Determining Stream Availability):

- `kinesis:ListShards` - Required by default to use the ListShards API. This API offers higher rate limits for shard verification.
- `kinesis:DescribeStream` - Required instead of `kinesis:ListShards` if the **ListShards API** advanced setting is toggled off.

Data Writing (Sending Data to the Stream):

- `kinesis:PutRecord` - Required for Gzip compression or batching (when **Advanced Settings** > **Compression** is set to `Gzip` or **Advanced Settings** > **Send batched** is toggled on).
- `kinesis:PutRecords` - Required for non-batched mode (when **Advanced Settings** > **Send batched** is toggled off).

Configure Cribl Edge to Output to Amazon Kinesis Data Streams

In Cribl Edge, configure Amazon CloudWatch Kinesis Data Streams:

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this Kinesis definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.

- **Description**: Optionally, enter a description.
- **Stream name**: Enter the name of the Kinesis Data Stream (not ARN) to which to send events.
- **Region**: Select the AWS Region where your Kinesis Data Stream is located. If your Region isn't listed, you can manually enter it. Additionally, you can specify the service endpoint in the **Endpoint** field under Advanced Settings.

3. Next, you can configure the following **Optional Settings**:
   - **Backpressure behavior**: Select whether to block, drop, or queue events when all receivers are exerting backpressure. (Causes might include a broken or denied connection, or a rate limiter.) Defaults to `Block`. When this field is set to `queue`, see Persistent Queue Settings for details.
   - **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
4. Optionally, configure any Authentication, Assume Role, Processing, and Advanced settings outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

> On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.
>
> This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.



If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5 GB`. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10

seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.



Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Authentication

Use the **Authentication Method** drop-down to select an AWS authentication method.

Auto: This default option uses the AWS SDK for JavaScript to automatically obtain credentials in the following order of attempts:

1. **Environment Variables**: Loaded from environment variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`.
2. **IAM Identity Center (SSO)**: Loaded from AWS IAM Identity Center configuration.
3. **Shared Credentials File**: Loaded from the shared credentials file (`~/.aws/credentials`).
4. **IAM Roles for Amazon EC2/ECS**: Loaded from AWS Identity and Access Management (IAM) roles attached to an EC2 instance or ECS container.
5. **JSON File on Disk**: Loaded from a JSON file on disk.
6. **Other Credential-Provider Classes**: Other credential-provider classes provided by the AWS SDK for JavaScript.

The `Auto` method works both when running on AWS and in other environments where the necessary credentials are available through one of the above methods.



SSO Providers

When using the auto authentication method, you can leverage SSO providers like SAML and Okta to issue temporary credentials. These credentials should be set in the environment variables

> `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`. The AWS SDK will then use these environment variables to authenticate.

For details, see AWS' Actions, resources, and condition keys for Amazon Kinesis Data Streams documentation.

Manual: If not running on AWS, you can select this option to enter a static set of user-associated IAM credentials (your access key and secret key) directly or by reference. This is useful for Workers not in an AWS VPC, for example, those running a private cloud. The **Manual** option exposes these corresponding additional fields:

- **Access key**: Enter your AWS access key. If not present, will fall back to the `env.AWS_ACCESS_KEY_ID` environment variable, or to the metadata endpoint for IAM role credentials.
- **Secret key**: Enter your AWS secret key. If not present, will fall back to the `env.AWS_SECRET_ACCESS_KEY` environment variable, or to the metadata endpoint for IAM credentials.

Secret: If not running on AWS, you can select this option to supply a stored secret that references an AWS access key and secret key. The **Secret** option exposes this additional field:

- **Secret key pair**: Use the drop-down to select an API key/secret key pair that you've configured in Cribl Edge's secrets manager. A **Create** link is available to store a new, reusable secret.

Assume Role

When using **Assume Role** to access resources in a different Region than Cribl Edge, you can target the AWS Security Token Service (STS) endpoint specific to that Region by using the `CRIBL_AWS_STS_REGION` environment variable on your Edge Node. To specify a custom STS endpoint, set the `CRIBL_AWS_STS_ENDPOINT` environment variable. If both are set, `CRIBL_AWS_STS_ENDPOINT` takes precedence. Setting an invalid Region or endpoint results in a fallback to the global STS endpoint.

Enable for Kinesis Streams: Toggle on to use Assume Role credentials to access Kinesis Streams.

AssumeRole ARN: Enter the Amazon Resource Name (ARN) of the role to assume.

External ID: Enter the External ID to use when assuming role.

Duration (seconds): Duration of the Assumed Role's session, in seconds. Minimum is 900 (15 minutes). Maximum is 43200 (12 hours). Defaults to 3600 (1 hour).

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options

include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Advanced Settings

Endpoint: Kinesis Stream service endpoint. If empty, the endpoint will be automatically constructed from the Region.



To proxy outbound HTTP/S requests, see System Proxy Configuration.

Signature version: Signature version to use for signing Kinesis Stream requests. Defaults to `v4`.

Put request concurrency: Maximum number of ongoing put requests before blocking. Defaults to `5`.

Record size limit (KB, uncompressed): Maximum size of each individual record before compression. For non-compressible data, 1MB (the default) is the maximum recommended size.

Flush period (sec): Maximum time between requests. Low settings could cause the payload size to be smaller than its configured maximum.

Compression: Type of compression to use on records being sent downstream. Defaults to `Gzip`, and when set to `None`, displays the following control:

Send batched: Whether to send multiple events batched as a single NDJSON Kinesis record (the default) or send each event as its own Kinesis record, serialized as JSON. When toggled off, displays the following control:

- **Records-per-flush limit**: Maximum number of records to send in a single request to the Amazon Kinesis Data Streams Service PutRecords API endpoint.

ListShards API: The ListShards API is used by default for Kinesis Stream shard verification. This provides significantly higher rate limits (1,000 calls/second/stream), minimizing throttling and improving data delivery speed and reliability, particularly in high-throughput environments.

- Using ListShards requires your IAM policy to have the `kinesis:ListShards` action to grant access. See Controlling Access to Amazon Kinesis Data Streams Resources Using IAM.
- Toggle off to use the DescribeStream API, which has lower rate limits (10 calls/second/account) and increased susceptibility to throttling.

Reuse connections: Whether to reuse connections between requests. Toggling on (default) can improve performance.

Reject unauthorized certificates: Whether to accept certificates that cannot be verified against a valid Certificate Authority (for example, self-signed certificates). Defaults to toggled on.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Record Formats in Detail

When sending multiple events within one record, the Record format will be as follows:

- A header line provides information about the payload (the one supported type is shown below).
- Each Kinesis record will contain multiple events serialized as Newline-Delimited JSON (NDJSON).
- Record payloads (including header and body) will be gzip-compressed if **Advanced Settings** > **Compression** is set to `Gzip`.

Sample Kinesis Record

```
{"format":"ndjson","count":8,"size":3960}
{"_raw":"07-03-2018 18:33:51.136 -0700 ERROR TcpOutputFd - Read error. Connection res
{"_raw":"07-03-2018 18:33:51.136 -0700 INFO  TcpOutputProc - Connection to 127.0.0.1:
...
```

Copy

When sending one event per Record, the Record format will be as follows:

- No header will be included.
- Each Kinesis record will contain a single event serialized as JSON (**not** NDJSON).
- Record payloads will not be compressed.

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

12.3.3.3. Amazon S3 Compatible Stores Destination

S3 is a non-streaming Destination type. Cribl Edge does **not** have to run on AWS to deliver data to S3.



Type: Non-Streaming | TLS Support: Yes | PQ Support: No

Stores that are S3-compatible will also work with this Destination type. For example, the S3 Destination can be adapted to send data to Snowflake, for which Cribl Edge currently has no preconfigured Destination.

For guidance in configuring these adaptations, contact Cribl Support. For general guidance, see our Amazon S3 Better Practices guide. To replay data you've previously exported to S3-compatible stores, see our Using S3 Storage and Replay guide and our interactive Data Collection & Replay sandbox.

Configure Cribl Edge to Output to S3 Destinations

In Cribl Edge, configure Amazon S3:

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this S3 definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
    - **Description**: Optionally, enter a description.
    - **S3 bucket name**: Name of the destination S3 Bucket. This value can be a constant, or a JavaScript expression that will be evaluated only at init time. For example, referencing a Global Variable: `myBucket-${C.vars.myVar}`.

        Event-level variables are not available for JavaScript expressions. This is because the bucket name is evaluated only at Destination initialization. If you want to use event-level variables in file paths, Cribl recommends specifying them in the **Partitioning Expression** field (described below), because this is evaluated for each file.

- **Staging location**: Filesystem location in which to locally buffer files before compressing and moving to final destination. Cribl recommends that this location be stable and high-performance.

  The **Staging location** field is not displayed or available on Cribl.Cloud-managed Edge Nodes.

- **Key prefix**: Root directory to prepend to path before uploading. Enter either a constant, or a JS expression (enclosed in single quotes, double quotes, or backticks) that will be evaluated only at init time.
- **Data format**: The output data format defaults to `JSON`. `Raw` and `Parquet` are also available. Selecting `Parquet` (supported only on Linux, not Windows) exposes a **Parquet Settings** left tab, where you **must** configure certain options in order to export data in Parquet format.

3. Next, you can configure the following **Optional Settings**:

   - **Region**: Region where the S3 bucket is located.

     You must grant your IAM role or user the `GetBucketLocation` permission to the S3 bucket when no **Region** is selected. See Amazon S3 Permissions for an example.

   - **Partitioning expression**: JavaScript expression that defines how files are partitioned and organized. Default is date-based. If blank, Cribl Edge will fall back to the event's `__partition` field value (if present); or otherwise to the root directory of the **Output Location** and **Staging Location**.
   - **Compression**: Data compression format used before moving to final destination. Defaults to `gzip` (recommended). This setting is not available when **Data format** is set to `Parquet`.
   - **File name prefix expression**: The output filename prefix. Must be a JavaScript expression (which can evaluate to a constant), enclosed in quotes or backticks. Defaults to `CriblOut`.
   - **File name suffix expression**: The output filename suffix. Must be a JavaScript expression (which can evaluate to a constant), enclosed in quotes or backticks. Defaults to `` `.${C.env["CRIBL_WORKER_ID"]}.${__format}${__compression === "gzip" ? ".gz" : ""}` ``, where `__format` can be `json` or `raw`, and `__compression` can be `none` or `gzip`.

     To prevent files from being overwritten, Cribl appends a random sequence of 6 characters to the end of their names. This also applies to prefix and suffix expressions in file names.

     For example, if you set the **File name prefix expression** to `CriblExec` and set the **File name suffix expression** to `.csv`, the file name might display as `CriblExec-adPRWM.csv` with `adPRWM` appended.

   - **Backpressure behavior**: Select whether to block or drop events when all receivers are exerting backpressure. (Causes might include an accumulation of too many files needing to be closed.) Defaults to `Block`.

    - **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

4. Optionally, configure any Authentication, Assume Role, Processing, Parquet, and Advanced settings outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

Authentication

Use the **Authentication method** drop-down to select one of these options:

Auto: This default option uses the AWS SDK for JavaScript to automatically obtain credentials in the following order of attempts:

1. **Environment Variables**: Loaded from environment variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`.
2. **IAM Identity Center (SSO)**: Loaded from AWS IAM Identity Center configuration.
3. **Shared Credentials File**: Loaded from the shared credentials file (`~/.aws/credentials`).
4. **IAM Roles for Amazon EC2/ECS**: Loaded from AWS Identity and Access Management (IAM) roles attached to an EC2 instance or ECS container.
5. **JSON File on Disk**: Loaded from a JSON file on disk.
6. **Other Credential-Provider Classes**: Other credential-provider classes provided by the AWS SDK for JavaScript.

The `Auto` method works both when running on AWS and in other environments where the necessary credentials are available through one of the above methods.



SSO Providers

When using the auto authentication method, you can leverage SSO providers like SAML and Okta to issue temporary credentials. These credentials should be set in the environment variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`. The AWS SDK will then use these environment variables to authenticate.

Manual: If not running on AWS, you can select this option to enter a static set of user-associated IAM credentials (your access key and secret key) directly or by reference. This is useful for Workers not in an AWS VPC, for example, those running a private cloud. The **Manual** option exposes these corresponding additional fields:

- **Access key**: Enter your AWS access key. If not present, will fall back to the `env.AWS_ACCESS_KEY_ID` environment variable, or to the metadata endpoint for IAM role credentials.

- **Secret key**: Enter your AWS secret key. If not present, will fall back to the `env.AWS_SECRET_ACCESS_KEY` environment variable, or to the metadata endpoint for IAM credentials.

The values for **Access key** and **Secret key** can be a constant or a JavaScript expression (such as `${C.env.MY_VAR}`).



Enclose values that contain special characters (like `/`) or environment variables in single quotes or backticks.

Secret: If not running on AWS, you can select this option to supply a stored secret that references an AWS access key and secret key. The **Secret** option exposes this additional field:

- **Secret key pair**: Use the drop-down to select an API key/secret key pair that you've configured in Cribl Edge's secrets manager. A **Create** link is available to store a new, reusable secret.

Assume Role

When using **Assume Role** to access resources in a different Region than Cribl Edge, you can target the AWS Security Token Service (STS) endpoint specific to that Region by using the `CRIBL_AWS_STS_REGION` environment variable on your Edge Node. To specify a custom STS endpoint, set the `CRIBL_AWS_STS_ENDPOINT` environment variable. If both are set, `CRIBL_AWS_STS_ENDPOINT` takes precedence. Setting an invalid Region or endpoint results in a fallback to the global STS endpoint.

Enable for S3: Toggle on to use Assume Role credentials to access S3.

AssumeRole ARN: Enter the Amazon Resource Name (ARN) of the role to assume.

External ID: Enter the External ID to use when assuming role. This is required only when assuming a role that requires this ID in order to delegate third-party access. For details, see AWS' documentation.

Duration (seconds): Duration of the Assumed Role's session, in seconds. Minimum is 900 (15 minutes). Maximum is 43200 (12 hours). Defaults to 3600 (1 hour).

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports `c*` wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.

- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Parquet Settings

To write out Parquet files, note that:

- On Linux, you can use the Cribl Edge CLI's `parquet` command to view a Parquet file, its metadata, or its schema.
- Cribl Edge Workers support Parquet only when running on Linux, not on Windows.
- See Parquet Schemas for pointers on how to avoid problems such as data mismatches.
- In Cribl Edge 4.3 and newer, the S3 Collector supports ingesting data in the Parquet format. Therefore, data that you export in Parquet format can be replayed via the Collector.

Automatic schema: Toggle on to automatically generate a Parquet schema based on the events of each Parquet file that Cribl Edge writes. When toggled off (the default), exposes the following additional field:

- **Parquet schema**: Select a schema from the drop-down.



If you need to modify a schema or add a new one, follow the instructions in our Parquet Schemas topic. These steps will propagate the freshest schema back to this drop-down.

Parquet version: Determines which data types are supported, and how they are represented. Defaults to `2.6`; `2.4` and `1.0` are also available.

Data page version: Serialization format for data pages. Defaults to `V2`. If your toolchain includes a Parquet reader that does not support `V2`, use `V1`.

Group row limit: The number of rows that every group will contain. The final group can contain a smaller number of rows. Defaults to `10000`.

Page size: Set the target memory size for page segments. Generally, set lower values to improve reading speed, or set higher values to improve compression. Value must be a positive integer smaller than the **Row group size** value, with appropriate units. Defaults to `1 MB`.

Log invalid rows: Toggle on to output up to 20 unique rows that were skipped due to data format mismatch. Log level must be set to `debug` for output to be visible.

Write statistics: Leave toggled on (the default) if you have Parquet tools configured to view statistics - these profile an entire file in terms of minimum/maximum values within data, numbers of nulls, etc.

Write page indexes: Leave toggled on (the default) if your Parquet reader uses statistics from Page Indexes to enable page skipping. One Page Index contains statistics for one data page.

Write page checksum: Toggle on if you have configured Parquet tools to verify data integrity using the checksums of Parquet pages.

Metadata (optional): The metadata of files the Destination writes will include the properties you add here as key-value pairs. For example, one way to tag events as belonging to the OCSF category for security findings would be to set **Key** to `OCSF Event Class` and **Value** to `2001`.

Advanced Settings

File size limit (MB): Maximum uncompressed output file size. Files of this size will be closed and moved to final output location. Defaults to `32`.

File open time limit (sec): Maximum amount of time to write to a file. Files open for longer than this limit will be closed and moved to final output location. Defaults to `300`.

Idle time limit (sec): Maximum amount of time to keep inactive files open. Files open for longer than this limit will be closed and moved to final output location. Defaults to `30`.

Open file limit: Maximum number of files to keep open concurrently. When this limit is exceeded, on any individual Worker Process, Cribl Edge will close the oldest open files, and move them to the final output location. Defaults to `100`.



Cribl Edge will close files when **any** of the four above conditions is met.

Compression level: Specifies the level of compression to apply before moving files to the final destination. Higher compression levels reduce file size but increase CPU usage. Choose from `Best Speed`, `Normal`, or `Best Compression`. The default level, `Best Speed`, will prioritize faster compression over smaller file sizes.

Staging file limit: Maximum number of files that the Destination will allow to wait for upload before it applies backpressure. Defaults to `100`; minimum is `10`; maximum is `4200`. For context, see Troubleshooting below.

Concurrent file parts limit: Maximum number of parts to upload in parallel per file. A value of `1` tells the Destination to send the whole file at once. When set to `2` or above, IAM permissions must include those required for multipart uploads. Defaults to `4`; highest allowed value is `10`.

Disk space protection: Specifies whether to `Block` (default) or `Drop` incoming events when the disk space falls below the globally defined **Min free disk space** amount.

Add Output ID: When toggled on (default), adds the **Output ID** field's value to the staging location's file path. This ensures that each Destination's logs will write to its own bucket.

For a Destination originally configured in a Cribl Edge version below 2.4.0, the **Add Output ID** behavior will be switched **off** on the backend, regardless of this toggle's state. This is to avoid losing any files pending in the original staging directory, upon Cribl Edge upgrade and restart. To enable this option for such Destinations, Cribl's recommended migration path is:

- Clone the Destination.
- Redirect the Routes referencing the original Destination to instead reference the new, cloned Destination.

This way, the original Destination will process pending files (after an idle timeout), and the new, cloned Destination will process newly arriving events with **Add output ID** enabled.

Remove empty staging directories: When toggled on (the default), Cribl Edge deletes empty staging directories after moving files. This prevents the proliferation of orphaned empty directories. When enabled, exposes this additional option:

- **Staging cleanup period**: How often (in seconds) to delete empty directories when **Remove staging dirs** is enabled. Defaults to `300` seconds (every 5 minutes). Minimum configurable interval is `10` seconds; maximum is `86400` seconds (every 24 hours).
- **Directory batch size**: Specifies how many directories are scanned per batch during automatic cleanup of empty staging directories. Set between `10` and `10000`, defaults to `1000`. Higher values speed up cleanup but increase memory usage.

Force close on shutdown: Toggle on to force all staged files to close during an orderly shutdown. This triggers immediate upload or finalization of any in-progress files, regardless of **Idle time**, **File open time**, or **File size** limits. Use this option to minimize data loss risk in ephemeral or auto-scaling environments. This could result in a higher number of small files, as files are closed before reaching their configured thresholds. Ensure your Worker Group or Fleet **Drain timeout** is set high enough to allow all staged data to be finalized before shutdown completes.

Enable dead-lettering: Toggle this on to set a maximum number of retries, and to move files to a designated directory when write failures exceed that limit. This prevents data flow blockage and excessive error logging due to undeliverable files. When enabled, exposes two additional fields:

- **Dead-letter location**: Specify the storage location for undeliverable files. Defaults to `$CRIBL_HOME/state/outputs/dead-letter`.
- **Maximum retry limit**: Configure the retry limit for failed file deliveries. This setting defines how many times the system will attempt to move a file to its intended location before it is deemed undeliverable and placed in the dead-letter directory. Defaults to `20`.

Endpoint: S3 service endpoint. If empty, Cribl Edge will automatically construct the endpoint from the AWS Region. To access the AWS S3 endpoints, use the path-style URL format. You don't need to specify the bucket

name in the URL, because Cribl Edge will automatically add it to the URL path. For details, see AWS' Path-Style Requests topic.

Object ACL: Object ACL (Access Control List) to assign to uploaded objects.

Storage class: Select a storage class for uploaded objects. Defaults to `Standard`. The other options are: `Reduced Redundancy Storage`; `Standard, Infrequent Access`; `One Zone, Infrequent Access`; `Intelligent Tiering`; `Glacier Flexible Retrieval`; `Glacier Instant Retrieval`; or `Glacier Deep Archive`.

Server-side encryption: Encryption type for uploaded objects - used to enable encryption on data at rest. Defaults to no encryption; the other options are `Amazon S3 Managed Key` or `AWS KMS Managed Key`.

> AWS S3 always encrypts data in transit using HTTPS, with default one-way authentication from server to clients. With other S3-compatible stores (such as our native MinIO Destination), use an `https://` URL to invoke in-transit encryption. Two-way authentication is not required to get encryption, and requires clients to possess a certificate.

Signature version: Signature version to use for signing S3 requests. Defaults to `v4`.

Reuse connections: Whether to reuse connections between requests. Toggling on (default) can improve performance.

Reject unauthorized certificates: Whether to accept certificates that cannot be verified against a valid Certificate Authority (for example, self-signed certificates). Defaults to toggled on.

Verify if bucket exists: Toggle off if you can access files within the bucket, but not the bucket itself. This resolves errors of the form: `error initializing...Bucket does not exist`. For context, see Troubleshooting below.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

IAM Permissions

The following permissions are needed to write to an Amazon S3 bucket:

- `s3:GetBucketLocation` - Needed if **Region** is not configured.
- `s3:ListBucket` - Needed unless **Verify if bucket exists** is toggled off.
- `s3:PutObject`

If your Destination needs to do multipart uploads to S3, two more permissions are needed:

- `kms:GenerateDataKey`

- `kms:Decrypt`

See the AWS documentation.

Temporary Access via SSO Provider

You can use Okta or SAML to provide access to S3 buckets using temporary security credentials.

Internal Fields

Cribl Edge uses a set of internal fields to assist in forwarding data to a Destination.

Field for this Destination:

- `__partition`

Proxying Requests

If you need to proxy HTTP/S requests, see System Proxy Configuration.

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

Cribl University offers an Advanced Troubleshooting > Destination Integrations: S3 short course. To follow the direct course link, first log into your Cribl University account. (To create an account, select the **Sign up** link. You'll need to click through a short **Terms & Conditions** presentation, with chill music, before proceeding to courses - but Cribl's training is always free of charge.) Once logged in, check out other useful Advanced Troubleshooting short courses and Troubleshooting Criblets.

Common Issues

“Bucket does not exist - self signed certificate”

Example Error Text:

```
message: Bucket does not exist - self signed certificate
stack: Error: self signed certificate
    at TLSSocket.onConnectSecure (node:_tls_wrap_1535:34)
    at TLSSocket.emit (node:events:513:28)
    at TLSSocket.emit (node:events:489:12)
    at TLSSocket._fini... Show more
tmpPath: /opt/cribl/s3_bucket/cribl/2023/03/28/CriblOut-XTNGh5.0.json.tmp
```

Copy

Can occur when:

- the Destination uses a self-signed cert that has not yet been trusted by Cribl Stream.
- the user does not have appropriate permissions to view the bucket.
- the authentication token generated by your computer has a timestamp that is out of sync with the server’s time, resulting in an authentication failure.

Recommendation

Disable **Advanced Settings** > **Reject unauthorized certificates** or; get the sender’s cert and add it to the `NODE_EXTRA_CA_CERTS` path for validation; or verify that the user has appropriate permissions.

Misconfigured or Nonexistent Buckets

If the S3 bucket that the Destination is trying to send files to does not exist or is misconfigured, the Destination logs will reflect the following pattern:

1. Upon initialization, the Destination will log an error that says the bucket cannot be found. This error will not be logged if **Advanced Settings** > **Verify if bucket exists** is turned off.
2. Every time the Destination “rolls” a new file out of staging to try to upload it to S3, the Destination will log an error, until the number of files rolled exceeds the value of **Advanced Settings** > **Max files to stage**.

Once the Destination has begun to apply backpressure:

- The Destination will log a backpressure warning.
- Events will stop flowing through the Destination.
- The Destination will stop creating staging files.
- Although the Destination will periodically attempt to upload the staged files, it will not log any additional failures.

12.3.3.4. Amazon SQS Destination

Cribl Edge supports sending events to Amazon Simple Queuing Service.



Type: Streaming | TLS Support: Yes | PQ Support: Yes

Configure Cribl Edge to Send Data to Amazon SQS

In Cribl Edge, configure Amazon SQS:

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this SQS Destination.
    - **Description**: Optionally, enter a description.
    - **Queue name**: The name, URL, or ARN of the SQS queue to send events to. This value must be a JavaScript expression (which can evaluate to a constant), enclosed in single quotes, double quotes, or backticks. To specify a non-AWS URL, use the format: `'{url}/<queueName>'`. (For example, `':port/<myQueueName>'`.)
    - **Queue type**: The queue type used (or created). Defaults to `Standard`. `FIFO` (First In, First Out) is the other option.
3. Next, you can configure the following **Optional Settings**:
    - **Message group ID**: This parameter applies only to queues of type FIFO. Enter the tag that specifies that a message belongs to a specific message group. (Messages belonging to the same message group are processed in FIFO order.) Defaults to `cribl`. Use event field `__messageGroupId` to override this value.
    - **Create queue**: Specifies whether to create the queue if it does not exist. Defaults to toggled on.
    - **Region**: Region where SQS queue is located.

- **Backpressure behavior**: Select whether to block, drop, or queue events when all receivers are exerting backpressure. (Causes might include a broken or denied connection, or a rate limiter.) Defaults to `Block`. When this field is set to `queue`, see Persistent Queue Settings for details.
- **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

4. Optionally, configure any Authentication, Assume Role, Processing, and Advanced settings outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

> On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.
>
> This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.

Mode	Description
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.



If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5` GB. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.



Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Authentication

Use the **Authentication method** drop-down to select an AWS authentication method.

Auto: This default option uses the AWS SDK for JavaScript to automatically obtain credentials in the following order of attempts:

1. **Environment Variables**: Loaded from environment variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`.
2. **IAM Identity Center (SSO)**: Loaded from AWS IAM Identity Center configuration.
3. **Shared Credentials File**: Loaded from the shared credentials file (`~/.aws/credentials`).
4. **IAM Roles for Amazon EC2/ECS**: Loaded from AWS Identity and Access Management (IAM) roles attached to an EC2 instance or ECS container.
5. **JSON File on Disk**: Loaded from a JSON file on disk.
6. **Other Credential-Provider Classes**: Other credential-provider classes provided by the AWS SDK for JavaScript.

The `Auto` method works both when running on AWS and in other environments where the necessary credentials are available through one of the above methods.



SSO Providers

When using the auto authentication method, you can leverage SSO providers like SAML and Okta to issue temporary credentials. These credentials should be set in the environment variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`. The AWS SDK will then use these environment variables to authenticate.

Manual: If not running on AWS, you can select this option to enter a static set of user-associated IAM credentials (your access key and secret key) directly or by reference. This is useful for Workers not in an AWS VPC, for example, those running a private cloud. The **Manual** option exposes these corresponding additional fields:

- **Access key**: Enter your AWS access key. If not present, will fall back to the `env.AWS_ACCESS_KEY_ID` environment variable, or to the metadata endpoint for IAM role credentials.
- **Secret key**: Enter your AWS secret key. If not present, will fall back to the `env.AWS_SECRET_ACCESS_KEY` environment variable, or to the metadata endpoint for IAM credentials.

Secret: If not running on AWS, you can select this option to supply a stored secret that references an AWS access key and secret key. The **Secret** option exposes this additional field:

- **Secret key pair**: Use the drop-down to select an API key/secret key pair that you've configured in Cribl Edge's secrets manager. A **Create** link is available to store a new, reusable secret.

Assume Role

When using **Assume Role** to access resources in a different Region than Cribl Edge, you can target the AWS Security Token Service (STS) endpoint specific to that Region by using the `CRIBL_AWS_STS_REGION` environment variable on your Edge Node. To specify a custom STS endpoint, set the `CRIBL_AWS_STS_ENDPOINT` environment variable. If both are set, `CRIBL_AWS_STS_ENDPOINT` takes precedence. Setting an invalid Region or endpoint results in a fallback to the global STS endpoint.

Enable for SQS: Toggle on to use Assume Role credentials to access SQS.

AWS account ID: Enter the SQS queue owner's AWS account ID. Leave empty if the SQS queue is in the same AWS account where this Cribl Edge instance is located.

AssumeRole ARN: Enter the Amazon Resource Name (ARN) of the role to assume.

External ID: Enter the External ID to use when assuming role.

Duration (seconds): Duration of the Assumed Role's session, in seconds. Minimum is 900 (15 minutes). Maximum is 43200 (12 hours). Defaults to 3600 (1 hour).

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.

- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Advanced Settings

Endpoint: SQS service endpoint. If empty, the endpoint will be automatically constructed from the Region.

Signature version: Signature version to use for signing SQS requests. Defaults to `v4`.

Queue size limit: Maximum number of queued batches before blocking. Defaults to `100`.

Record size limit (KB): Maximum size of each individual record. Per the SQS spec, the maximum allowed value is 256 KB. (the default).

Flush period (sec): Maximum time between requests. Low settings could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Concurrent request limit: The maximum number of in-progress API requests before backpressure is applied. Defaults to `10`.

Reuse connections: Whether to reuse connections between requests. Toggling on (default) can improve performance.

Reject unauthorized certificates: Whether to accept certificates that cannot be verified against a valid Certificate Authority (for example, self-signed certificates). Defaults to toggled on.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

SQS Permissions

The following permissions are needed to write to an SQS queue:

- `sqs:ListQueues`
- `sqs:SendMessage`
- `sqs:SendMessageBatch`
- `sqs:CreateQueue`
- `sqs:GetQueueAttributes`
- `sqs:SetQueueAttributes`

- `sqs:GetQueueUrl`

Internal Fields

Cribl Edge uses a set of internal fields to assist in handling of data. These "meta" fields are **not** part of an event, but they are accessible, and functions can use them to make processing decisions.

Fields for this Destination:

- `__messageGroupId`
- `__sqsMsgAttrs`
- `__sqsSysAttrs`

Proxying Requests

If you need to proxy HTTP/S requests, see System Proxy Configuration.

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

12.3.3.5. Amazon Security Lake Destination

This Destination delivers data to Amazon Security Lake, a fully-managed security data lake service backed by Amazon S3 buckets.

Type: Non-Streaming | TLS Support: Yes | PQ Support: No

This Destination is unavailable on Windows and macOS.

Security Lake ingests events described by the Open Cybersecurity Schema Framework (OCSF) and conveyed as Parquet files. This Destination sends the Parquet files in batches. Cribl Edge does not read or replay the data sent to Security Lake.

A Complete Setup Scenario

Cribl recommends that you start with the Amazon Security Lake Integration topic, which will guide you through the following steps in the correct order:

1. Set up Amazon Security Lake.
2. Set up Cribl Stream.
3. Create an Amazon Security Lake custom source, This is an S3 bucket dedicated to a single OCSF event class.
4. Create a Cribl Amazon Security Lake Destination.
5. Connect a Cribl Source to the Amazon Security Lake Destination.
6. Send data to Amazon Security Lake.

As outlined above, you'll create and configure this Destination as the fourth of the six overall steps. The Integration topic covers the key points about configuring this Destination. Refer back to this reference topic, as needed, for details about any given setting. Some of the instructions below assume that you have recorded IDs, values, etc., from procedures you've already completed according to the Integration topic.

Although Cribl Edge does **not** need to be deployed in Cribl.Cloud in order to deliver data to Amazon Security Lake, setup procedures on Cribl>Cloud will differ from those documented here. To learn more, please contact Cribl via the `#packs` channel of Cribl Community Slack.

Configure Cribl Edge to Output to Amazon Security Lake

In Cribl Edge, configure Amazon Security Lake:

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**: -**Output ID**: Enter a unique name to identify this Amazon Security Lake Destination definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
    - **Description**: Optionally, enter a description.
    - **S3 bucket name**: Name of the destination S3 Bucket you created when setting up Amazon Security Lake. This value can be a constant, or a JavaScript expression that will be evaluated only at init time. For example, referencing a Global Variable: `myBucket-${C.vars.myVar}`.

      Event-level variables are not available for JavaScript expressions. This is because the bucket name is evaluated only at Destination initialization.
    - **Region**: Region where the Amazon Security Lake is located.
    - **Account ID**: The ID of the AWS Account whose admin enabled Amazon Security Lake.
    - **Custom source name**: The name of the custom source you created in Amazon Security Lake to receive data from this Destination.
    - **Staging location**: Filesystem location in which to locally buffer files before compressing and moving to final destination. Cribl recommends that this location be stable and high-performance.

      The **Staging location** field is not displayed or available on Cribl.Cloud-managed Edge Nodes.
3. Next, you can configure the following **Optional Settings**:
    - **File name prefix expression**: The output filename prefix. Must be a JavaScript expression (which can evaluate to a constant), enclosed in quotes or backticks. Defaults to `CriblOut`.
    - **Backpressure behavior**: Select whether to block or drop events when all receivers are exerting backpressure. (Causes might include an accumulation of too many files needing to be closed.) Defaults to `Block`.
    - **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

4. Optionally, configure any Authentication, Assume Role, Processing, Parquet, and Advanced settings outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

Authentication

For the **Authentication Method**, you **must** leave the default **Auto** method selected.

This method uses the AWS instance's metadata service to automatically obtain short-lived credentials from the IAM role attached to an EC2 instance. The attached IAM role grants Cribl Edge Edge Nodes access to authorized AWS resources. Can also use the environment variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`. Works only when running on AWS.

Assume Role

For these settings, you'll enter values that you noted when creating the custom source in Security Lake.

AssumeRole ARN: The custom source's **Provider role ARN**.

External ID: In the custom source, this is the value of the `ExternalId` element of the trust relationship JSON object.

Duration (seconds): Duration of the Assumed Role's session, in seconds. Minimum is 900 (15 minutes). Maximum is 43200 (12 hours). Defaults to 3600 (1 hour).

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output. Use the drop-down to select the Pack you installed when setting up Cribl Edge.

System fields: Remove any fields specified here.

Parquet Settings

To write out Parquet files, note that:

- On Linux, you can use the Cribl Edge CLI's `parquet` command to view a Parquet file, its metadata, or its schema.
- Cribl Edge Workers support Parquet only when running on Linux, not on Windows.
- See Parquet Schemas for pointers on how to avoid problems such as data mismatches.
- In Cribl Edge 4.3 and newer, the S3 Collector supports ingesting data in the Parquet format. Therefore, data that you export in Parquet format can be replayed via the Collector.

Automatic schema: Toggle on to automatically generate a Parquet schema based on the events of each Parquet file that Cribl Edge writes. When toggled off (the default), exposes the following additional field:

Parquet schema: Select a schema from the drop-down.



If you need to modify a schema or add a new one, follow the instructions in our Parquet Schemas topic. These steps will propagate the freshest schema back to this drop-down.

Parquet version: Determines which data types are supported, and how they are represented. Defaults to `2.6`; `2.4` and `1.0` are also available.

Data page version: Serialization format for data pages. Defaults to `V2`. If your toolchain includes a Parquet reader that does not support `V2`, use `V1`.

Group row limit: The number of rows that every group will contain. The final group can contain a smaller number of rows. Defaults to `10000`.

Page size: Ideal memory size for page segments. For example, 1 MB or 128 MB. Generally, lower values improve reading speed, while higher values improve compression. Imposes a target, not a strict limit; the final size of a row group may be larger or smaller. Defaults to 1 MB.

Log invalid rows: Toggle on to output up to 20 unique rows that were skipped due to data format mismatch. Log level must be set to `debug` for output to be visible.

Write statistics: Leave toggled on (the default) if you have Parquet tools configured to view statistics – these profile an entire file in terms of minimum/maximum values within data, numbers of nulls, etc.

Write page indexes: Leave toggled on (the default) if your Parquet reader uses statistics from Page Indexes to enable page skipping. One Page Index contains statistics for one data page.

Write page checksum: Toggle on if you have configured Parquet tools to verify data integrity using the checksums of Parquet pages.

Metadata (optional): The metadata of files the Destination writes will include the properties you add here as key-value pairs. For example, one way to tag events as belonging to the OCSF category for security findings would be to set **Key** to `OCSF Event Class` and **Value** to `2001`.

Advanced Settings

File size limit (MB): Maximum uncompressed output file size. Files of this size will be closed and moved to final output location. Defaults to `32`.

File open time limit (sec): Maximum amount of time to write to a file. Files open for longer than this limit will be closed and moved to final output location. Defaults to `300`.

Idle time limit (sec): Maximum amount of time to keep inactive files open. Files open for longer than this limit will be closed and moved to final output location. Defaults to `30`.

Open file limit: Maximum number of files to keep open concurrently. When this limit is exceeded, on any individual Edge Node Process, Cribl Edge will close the oldest open files, and move them to the final output location. Defaults to `100`.



Cribl Edge will close files when **any** of the four above conditions is met.

Concurrent file parts limit: Maximum number of parts to upload in parallel per file. A value of `1` tells the Destination to send the whole file at once. When set to `2` or above, IAM permissions must include those required for multipart uploads. Defaults to `4`; highest allowed value is `10`.

Add Output ID: When toggled on (default), adds the **Output ID** field's value to the staging location's file path. This ensures that each Destination's logs will write to its own bucket.

Remove empty staging directories: When toggled on (the default), Cribl Edge deletes empty staging directories after moving files. This prevents the proliferation of orphaned empty directories. When enabled, exposes this additional option:

- **Staging cleanup period**: How often (in seconds) to delete empty directories when **Remove staging dirs** is enabled. Defaults to `300` seconds (every 5 minutes). Minimum configurable interval is `10` seconds; maximum is `86400` seconds (every 24 hours).
- **Directory batch size**: Specifies how many directories are scanned per batch during automatic cleanup of empty staging directories. Set between `10` and `10000`, defaults to `1000`. Higher values speed up cleanup but increase memory usage.

Enable dead-lettering: Toggle this on to set a maximum number of retries, and to move files to a designated directory when write failures exceed that limit. This prevents data flow blockage and excessive error logging due to undeliverable files. When enabled, exposes two additional fields:

- **Dead-letter location**: Specify the storage location for undeliverable files. Defaults to `$CRIBL_HOME/state/outputs/dead-letter`.
- **Maximum retry limit**: Configure the retry limit for failed file deliveries. This setting defines how many times the system will attempt to move a file to its intended location before it is deemed undeliverable and placed in the dead-letter directory. Defaults to `20`.

Endpoint: Security Lake service endpoint. If empty, defaults to AWS' Region-specific endpoint. Otherwise, it must point to Security Lake-compatible endpoint.

Object ACL: Object ACL (Access Control List) to assign to uploaded objects.

Storage class: Select a storage class for uploaded objects. Defaults to `Standard`. The other options are: `Reduced Redundancy Storage`; `Standard, Infrequent Access`; `One Zone, Infrequent Access`; `Intelligent Tiering`; `Glacier`; or `Deep Archive`.

Server-side encryption: Encryption type for uploaded objects - used to enable encryption on data at rest. Defaults to no encryption; the other options are `Amazon S3 Managed Key` or `AWS KMS Managed Key`.



Amazon S3 always encrypts data in transit using HTTPS, with default one-way authentication from server to clients. Two-way authentication is not required to get encryption, and requires clients to possess a certificate.

Signature version: Signature version to use for signing Security Lake requests. Defaults to `v4`.

Reuse connections: Whether to reuse connections between requests. Toggling on (default) can improve performance.

Reject unauthorized certificates: Whether to accept certificates that cannot be verified against a valid Certificate Authority (for example, self-signed certificates). Defaults to toggled on.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

IAM Permissions

The following permissions are always needed to write to an Amazon S3 bucket:

- `s3:ListBucket`
- `s3:GetBucketLocation`
- `s3:PutObject`

If your Destination needs to do multipart uploads to S3, two more permissions are needed:

- `kms:GenerateDataKey`
- `kms:Decrypt`

See the AWS documentation.

Creating and Modifying Parquet Schemas

When you need to add a new Parquet schema or modify an existing one:

1. Navigate to the Parquet Schema Library, located at **Processing** > **Knowledge** > **Parquet Schemas**.

2. Create or modify the schema, as described in Parquet Schemas.



Cribl strongly recommends that to modify an existing schema, you first clone it and give the clone its own distinct name.

3. Return to this Destination, and select the new or modified schema in the **Parquet schema** drop-down.
4. Click **Save**.

Cloning First is Safer Than Just Modifying

Modifying an existing schema in the Schema Library does **not** propagate your modifications to the Destination's copy of the schema.

- Cloning and renaming schemas is the safest approach, because in Step 3 above, it removes any doubt that your Destination will now use the newly-modified version of your schema.
- If you do **not** clone and rename the schema (i.e., you leave the schema name unchanged), you still must re-select the schema in the Destination's **Parquet schema** drop-down, and click **Save**, to bring the modified version into the Destination.

Internal Fields

Cribl Edge uses a set of internal fields to assist in forwarding data to a Destination.

Field for this Destination:

- `__partition`

Troubleshooting Resources



Cribl University offers an Advanced Troubleshooting > Destination Integrations: S3 short course. Since Amazon Security Lake relies on S3 buckets, what you learn from the course can augment your Security Lake skills.
To follow the direct course link, first log into your Cribl University account. (To create an account, select the **Sign up** link.
You'll need to click through a short **Terms & Conditions** presentation, with chill music, before proceeding to courses - but Cribl's training is always free of charge.)
Once logged in, check out other useful Advanced Troubleshooting short courses and Troubleshooting Criblets.

12.3.4. Azure

You can send data from Cribl Edge to the following Azure services:

Destination	Description
Azure Blob Storage	Deliver data to Azure Blog Storage or Azure Data Lake Storage Gen2
Azure Data Explorer	Send batched or streaming data to the Azure Data Explorer managed data analytics service
Azure Event Hubs	Send data to Azure Event Hubs
Azure Monitor Logs	Send data to Azure Monitor Logs
Microsoft Sentinel	Send Log and Metric Events to Microsoft Sentinel SIEM

12.3.4.1. Azure Blob Storage Destination

Cribl Edge supports sending data to (and replaying specific events from) both Azure Blob Storage, and Azure Data Lake Storage Gen2, which implements a hierarchical namespace over blob data. This Destination can deliver data to Azure whether Cribl Edge is running on Azure, another cloud platform, or on-prem.

To use Azure Data Lake Storage Gen2 with this Destination, configure an Azure Storage account with hierarchical namespace enabled and use the standard connection string format. No additional settings are required to target Azure Data Lake Storage Gen2.



Type: Non-Streaming | TLS Support: Yes | PQ Support: No

For configuration examples, see Resources below.

Configure Cribl Edge to Output to Azure Blob Storage

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this Destination definition.
    - **Description**: Optionally, enter a description.
    - **Container name**: Specify the Azure Blob Storage container name. Containers organize blobs, similar to directories in a file system. Container names can include only lowercase letters, numbers, and hyphens.
        - For dynamic container names, provide a JavaScript expression within quotes or backticks, for example, `` `myContainer-${C.env["CRIBL_WORKER_ID"]}` ``. This expression is evaluated at initialization, resolving environment or runtime variables. Only simple concatenations and references to global variables are supported.
    - **Blob prefix**: Root directory to prepend to path before uploading.

- **Staging location**: Local filesystem location in which to buffer files before compressing and moving them to the final destination. Cribl recommends that this location be stable and high-performance. Default is `$CRIBL_HOME/state/outputs/staging`.

  > In Cribl Stream, the **Staging location** field is not displayed or available on Cribl.Cloud-managed Worker Groups.

- **Data format**: The output data format defaults to `JSON`. `Raw` and `Parquet` are also available. Selecting `Parquet` (supported only on Linux, not Windows) exposes a **Parquet Settings** left tab, where you **must** configure certain options in order to export data in Parquet format.

3. Under **Authentication**, select an **Authentication method** from the dropdown:
   - **Manual**: Use this default option to enter your Azure Storage connection string directly. Exposes a **Connection string** field for this purpose. (If left blank, Cribl Edge will fall back to `env.AZURE_STORAGE_CONNECTION_STRING`.)
   - **Secret**: This option exposes a **Connection string (text secret)** drop-down, in which you can select a stored secret that references an Azure Storage connection string. A **Create** link is available to store a new, reusable secret. For details, see Connection String Format.
   - **Client secret**: This option allows you to use the Azure service principal's client secret to authenticate, and exposes Azure service principal settings.
   - **Certificate**: This option allows you to use a certificate registered with the Azure service principal to authenticate, and exposes Azure service principal settings.
4. Next, you can configure the following **Optional Settings**:
   - **Create container**: Toggle on to create the configured container in Azure Blob Storage if one does not already exist. Hierarchical namespaces are typically used when working with Azure Data Lake Storage Gen2. When the target storage account has hierarchical namespace enabled, any container created by this Destination will therefore be an ADLS Gen2 container by default.
   - **Partitioning expression**: JavaScript expression that defines how files are partitioned and organized. Default is date-based. If blank, Cribl Edge will fall back to the event's `__partition` field value (if present); or otherwise to the root directory of the **Output Location** and **Staging Location**.
   - **Compression**: Data compression format used before moving to final destination. Defaults to `gzip` (recommended). This setting is not available when **Data format** is set to `Parquet`.
   - **File name prefix expression**: The output file name prefix. Must be a JavaScript expression (which can evaluate to a constant), enclosed in quotes or backticks. Defaults to `CriblOut`.
   - **File name suffix expression**: The output file name suffix. Must be a JavaScript expression (which can evaluate to a constant), enclosed in quotes or backticks. Defaults to `` `.${C.env["CRIBL_WORKER_ID"]}.${__format}${__compression === "gzip" ? ".gz" : ""}` ``, where `__format` can be `json` or `raw`, and `__compression` can be `none` or `gzip`.

To prevent files from being overwritten, Cribl appends a random sequence of 6 characters to the end of their names. This also applies to prefix and suffix expressions in file names.

For example, if you set the **File name prefix expression** to `CriblExec` and set the **File name suffix expression** to `.csv`, the file name might display as `CriblExec-adPRWM.csv` with `adPRWM` appended.

- **Backpressure behavior**: Whether to block or drop events when all receivers are exerting backpressure. (Causes might include an accumulation of too many files needing to be closed.) Defaults to `Block`.
- **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

5. Optionally, you can adjust the Processing, Parquet Settings, and Advanced settings outlined in the sections below.
6. Select **Save**, then **Commit & Deploy**.

Connection String Format

Manual and **Secret** authentication methods use an Azure Storage connection string in this format:

```
DefaultEndpointsProtocol=[http|https];AccountName=<your-account-name>;AccountKey=<you
```

Copy

A fictitious example, using Microsoft's recommended HTTPS option, is:

```
DefaultEndpointsProtocol=https;AccountName=storagesample;AccountKey=12345678...32;Enc
```

Copy

For Azure Data Lake Storage Gen2 accounts, use the same connection string pattern as for standard Azure Blob Storage.

Service Principal Settings

Client secret and **Certificate** authentication methods expose the following additional settings:

- **Storage account name**: Enter the name of your Azure Storage Account.

- **Tenant ID**: Enter the service principal's tenant ID.
- **Client ID**: Enter the service principal's client ID.
- **Client secret (text secret)** (with **Client secret** method selected): Select a text secret containing the client secret, or create a new one.
- **Certificate** (with **Certificate** method selected): Select the certificate you registered as credentials for your app in the Azure portal, or create a new one.
- **Azure Cloud**: Select the Azure cloud environment to use, `Azure Public Cloud` (default), `Azure China`, or `Azure Government`. The **Endpoint suffix** advanced setting is available if you need to override the default values.
    - Selecting `Azure China` sets the authority host to `login.chinacloudapi.cn` and the endpoint suffix to `core.chinacloudapi.cn`.
    - Selecting `Azure Government` sets the authority host to `login.microsoftonline.us` and the endpoint suffix to `core.usgovcloudapi.net`.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Parquet Settings

To write out Parquet files, note that:

- On Linux, you can use the Cribl Edge CLI's `parquet` command to view a Parquet file, its metadata, or its schema.
- Cribl Edge Workers support Parquet only when running on Linux, not on Windows.
- See Parquet Schemas for pointers on how to avoid problems such as data mismatches.

Automatic schema: Toggle on to automatically generate a Parquet schema based on the events of each Parquet file that Cribl Edge writes. When toggled off (the default), exposes the following additional field:

- **Parquet schema**: Select a schema from the drop-down.



If you need to modify a schema or add a new one, follow the instructions in our Parquet Schemas topic. These steps will propagate the freshest schema back to this drop-down.

Parquet version: Determines which data types are supported, and how they are represented. Defaults to `2.6`; `2.4` and `1.0` are also available.

Data page version: Serialization format for data pages. Defaults to `V2`. If your toolchain includes a Parquet reader that does not support `V2`, use `V1`.

Group row limit: The number of rows that every group will contain. The final group can contain a smaller number of rows. Defaults to `10000`.

Page size: Set the target memory size for page segments. Generally, set lower values to improve reading speed, or set higher values to improve compression. Value must be a positive integer smaller than the **Row group size** value, with appropriate units. Defaults to `1 MB`.

Log invalid rows: Toggle on to output up to 20 unique rows that were skipped due to data format mismatch. Log level must be set to `debug` for output to be visible.

Write statistics: Leave toggled on (the default) if you have Parquet tools configured to view statistics - these profile an entire file in terms of minimum/maximum values within data, numbers of nulls, and so on.

Write page indexes: Leave toggled on (the default) if your Parquet reader uses statistics from Page Indexes to enable page skipping. One Page Index contains statistics for one data page.

Write page checksum: Toggle on if you have configured Parquet tools to verify data integrity using the checksums of Parquet pages.

Metadata (optional): The metadata of files the Destination writes will include the properties you add here as key-value pairs. For example, one way to tag events as belonging to the OCSF category for security findings would be to set **Key** to `OCSF Event Class` and **Value** to `2001`.

Advanced Settings

File size limit (MB): Maximum uncompressed output file size. Files reaching this size will be closed and moved to the final output location. Defaults to `32`.

File open time limit (sec): Maximum amount of time to write to a file. Files open for longer than this limit will be closed and moved to final output location. Defaults to `300`.

Idle time limit (sec): Maximum amount of time to keep inactive files open. Files open for longer than this limit will be closed and moved to final output location. Default: `30`.

Open file limit: Maximum number of files to keep open concurrently. When exceeded, the oldest open files will be closed and moved to final output location. Default: `100`.



Cribl Edge will close files when **either** of the `File size limit (MB)` or the `Max file open time (sec)` conditions are met.

Concurrent file parts limit: Maximum number of parts to upload in parallel per file. A value of `1` tells the Destination to send one part at a time - that is, to upload the file's contents sequentially. Defaults to `1`; highest allowed value is `10`.

Blob access tier: Select the access tier for storing your data in Azure Blob Storage. Defaults to `Default account access tier`. Options include:

- `Hot tier` for frequently accessed data.
- `Cool tier` for infrequent access.
- `Cold tier` for rarely accessed data.
- `Archive tier` for data that can tolerate retrieval delays.

Disk space protection: Specifies whether to `Block` (default) or `Drop` incoming events when the disk space falls below the globally defined Min free disk space amount.

Add Output ID: When toggled on (default), adds the **Output ID** field's value to the staging location's file path. This ensures that each Destination's logs will write to its own bucket.



For a Destination originally configured in a Cribl Edge version below 2.4.0, the **Add Output ID** behavior will be switched **off** on the backend, regardless of this toggle's state. This is to avoid losing any files pending in the original staging directory, upon Cribl Edge upgrade and restart. To enable this option for such Destinations, Cribl's recommended migration path is:

- Clone the Destination.
- Redirect the Routes referencing the original Destination to instead reference the new, cloned Destination.

This way, the original Destination will process pending files (after an idle timeout), and the new, cloned Destination will process newly arriving events with **Add output ID** enabled.

Remove empty staging directories: When toggled on (the default), Cribl Edge deletes empty staging directories after moving files. This prevents the proliferation of orphaned empty directories. When enabled, exposes this additional option:

- **Staging cleanup period**: How often (in seconds) to delete empty directories when **Remove staging dirs** is enabled. Defaults to `300` seconds (every 5 minutes). Minimum configurable interval is `10` seconds; maximum is `86400` seconds (every 24 hours).
- **Directory batch size**: Specifies how many directories are scanned per batch during automatic cleanup of empty staging directories. Set between `10` and `10000`, defaults to `1000`. Higher values speed up cleanup but increase memory usage.

Force close on shutdown: Toggle on to force all staged files to close during an orderly shutdown. This triggers immediate upload or finalization of any in-progress files, regardless of **Idle time**, **File open time**, or **File size** limits. Use this option to minimize data loss risk in ephemeral or auto-scaling environments. This could result in a higher number of small files, as files are closed before reaching their configured thresholds. Ensure your Worker Group or Fleet **Drain timeout** is set high enough to allow all staged data to be finalized before shutdown completes.

Enable dead-lettering: Toggle this on to set a maximum number of retries, and to move files to a designated directory when write failures exceed that limit. This prevents data flow blockage and excessive error logging due to undeliverable files. When enabled, exposes two additional fields:

- **Dead-letter location**: Specify the storage location for undeliverable files. Defaults to `$CRIBL_HOME/state/outputs/dead-letter`.
- **Maximum retry limit**: Configure the retry limit for failed file deliveries. This setting defines how many times the system will attempt to move a file to its intended location before it is deemed undeliverable and placed in the dead-letter directory. Defaults to `20`.

Endpoint suffix: (Optional) Overrides the default endpoint suffix set by the **Azure Cloud** setting. Use this only if you need to connect to a custom or non-standard Azure Blob Storage endpoint.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Azure Permissions

When configuring an Azure Destination in Cribl Edge, you must grant the necessary permissions for it to interact with your Azure environment. Cribl recommends creating an Azure service principal with the minimum required permissions using Azure role-based access control (Azure RBAC). This approach ensures Cribl Edge has a dedicated identity with precisely the access it needs, following the principle of least privilege.

The minimum permissions required for Destinations are:

- **Storage Blob Data Contributor**

See Permissions for calling data operations in the official Microsoft documentation for more information.

Internal Fields

Cribl Edge uses a set of internal fields to assist in forwarding data to a Destination.

Field for this Destination:

- `__partition`

Proxying Requests

If you need to proxy HTTP/S requests, see System Proxy Configuration.

Resources

For examples of configuring Cribl Edge to interoperate with Azure services, see these guides:

- Preparing the Azure Workspace.
- SSO with Microsoft Entra ID (Cribl.Cloud).
- SSO with Microsoft Entra ID (On-Prem).

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

12.3.4.2. Azure Data Explorer Destination

Cribl Edge supports sending data to the Azure Data Explorer (ADX) managed data analytics service; you can then run Kusto queries against the data. This Destination can deliver data to Azure whether Cribl Edge is running on Azure, another cloud platform, or on-prem.

ADX stores log data in databases, which in turn contain one or more of the tables defined in the Azure namespace. You must create a separate Cribl Edge ADX Destination for each table that will store your data.



Type: Non-Streaming **or** Streaming (configurable) | TLS Support: Yes | PQ Support: No

For configuration examples, see Resources below.

Prerequisites

Before you configure the Destination, you need to collect some information from your Azure deployment. Start from **Home** (`portal.azure.com`), then navigate to the locations described below and collect the items specified.

Information About Your ADX Cluster

From **Home**, navigate to:

Azure Data Explorer Clusters > `your_Azure_cluster_name` > **Overview** > **Essentials**

- Copy the **URI**. In Cribl Edge, this will be the value for **General Settings** > **Cluster base URI**. Make a copy of the URI with `/.default` appended - this will be the value of **Authentication Settings** > **Scope**.
- Copy the **Data Ingestion URI**. In Cribl Edge, this will be the value for **General Settings** > **Ingestion service URI**.

Information About Your Azure Database

From **Home**, navigate to:

Azure Data Explorer Clusters > `your_Azure_cluster_name` | **Databases** > `database_name` | **Query**

- Note the name of your desired target table - this table name will be the value of **General Settings** > **Table name**.

Information About Your Azure App

First, from **Home**, navigate to:

App registrations > `your_Azure_app_name` | **Overview** | **Essentials**

- Copy the **Application (client) ID** - this will be the value of **Authentication Settings** > **Client ID**.
- Copy the **Directory (tenant) ID** - this will be the value of **Authentication Settings** > **Tenant ID**.

Nex, from **Home**, navigate to:

App registrations > `your_Azure_app_name` | **Certificates & secrets**

Now what you do depends on which method for **Authentication** you plan to use in Cribl Edge.

To use the Client secret method

- In the **Client secrets** tab, click **New client secret** and copy the **Value** of the new secret. In Cribl Edge, this will be the value of **Client secret**.

To use the Client secret (text secret) method

- In the **Client secrets** tab, click **New client secret** and copy the **Value** of the new secret. In Cribl Edge, this will be the value of **Create new secret** > **Value**.

To use the Certificate method

After you have generated a certificate and keys (if you do this later on in the configuration process, return to this section then):

- In the **Certificates** tab, click **Upload Certificate**. In the resulting drawer, follow the prompts to upload your certificate, and click **Add** to close the drawer.

The chosen certificate will then appear in the **Certificates** tab's list.

Data Mapping Prerequisites

Finally, you might need to note or copy data mapping information to obtain values for some of the **General Settings**.

ADX uses a data mapping to map fields from incoming data to fields in the target table. To see the data mapping, you'll run a Kusto query against your target table.

Here's the query syntax:

```
.show table EntityName ingestion MappingKind mapping MappingName
```

Copy

Here's an example query:

```
.show table SyslogTable ingestion json mapping "SyslogMapping"
```

Copy

The Kusto query that originally created this mapping would look like this:

```
.create table SyslogTable ingestion json mapping "SyslogMapping"
```

Copy

In Cribl Edge, you can use the names of mappings like the above example in the **General Settings** > **Data mapping** field. Alternatively, you can create a mapping without naming it, then copy and paste it (as a JSON object) into the Cribl Edge UI. To do this, toggle **General Settings** > **Mapping object** on. This hides the default **Data mapping** text field and opens a text window that expects JSON.

Configure Cribl Edge to Output to Azure Data Explorer

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this Destination definition.
    - **Description**: Optionally, enter a description.
    - **Ingestion mode**: Use the buttons to select **Batching** mode (the default) or **Streaming** mode. For details, see Ingestion Mode.

- **Cluster base URI**: Enter the base URI for your ADX cluster.
- **Ingestion service URI**: Displayed only in **Batching** mode. Enter the Ingestion Service URI for your ADX cluster.
- **Database name**: Enter the name of the ADX database where the target table resides.
- **Table name**: Enter the name of the target table.
- **Validate database settings**: When you save or start the Destination, validates the database name and credentials that you have entered in these settings. Also validates the table name, except when **Add mapping object** (below) is on. Defaults to toggled on. Toggle off if your Azure app does not have **both** the **Database Viewer** and the **Table Viewer** role.
- **Add mapping object**: Displayed only in **Batching** mode, this control is toggled off by default. Toggle on when you want to paste in a data mapping as a JSON object, instead of using a named data mapping. See Data Mapping Prerequisites above.
- **Data mapping**: If **Add mapping object** is off, enter the name of the desired data mapping. If **Add mapping object** is on, enter the desired data mapping as a JSON object.
- **Compression**: By default, this Destination compresses data using `gzip`; otherwise it sends the data uncompressed. This setting is not available when **Data format** is set to `Parquet`.
- **Data format**: The output data format defaults to `JSON`. `Raw` and `Parquet` are also available. Selecting `Parquet` (available only in **Batching** mode, and supported only on Linux, not Windows) exposes a **Parquet Settings** tab, where you select the Parquet schema.
- **Backpressure behavior**: Whether to block or drop events when all receivers are exerting backpressure. (Causes might include an accumulation of too many files needing to be closed.) Defaults to `Block`. When **Ingestion mode** is set to **Streaming** mode, the **Persistent Queue** option becomes available. See **Persistent Queue Settings** below.
- **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

3. Optionally, you can adjust the Authentication, Persistent Queue , Processing, Parquet, and Advanced settings outlined in the sections below.
4. Select **Save**, then **Commit & Deploy**.

Ingestion Mode

This Destination supports two ingestion modes:

- **Batching** (the default) stages data in a storage container from which ADX pulls batches of data, then writes them to the target ADX table. Batching works well when the Destination needs to ingest large amounts of data in short periods of time.

- **Streaming** sends data as HTTP request payloads, directly to the target ADX table, without staging them in a container first. Streaming can achieve lower latency than batching, as long as the data that the Destination ingests arrives in relatively small amounts. When in this mode, the Retries section (left tab) becomes available.

Authentication Settings

Microsoft Entra ID authentication endpoint: From the drop-down, select the Microsoft Entra ID endpoint that provides authentication tokens. To understand the available choices, see the Microsoft Entra documentation.

Tenant ID: Directory ID (tenant identifier) in Microsoft Entra ID.

Client ID: `client_id` to pass in the OAuth request parameter.

Scope: Scope to pass in the OAuth request parameter.

Use the **Authentication method** buttons to select one of these options:

- **Client secret**: Use this default option to enter the client secret that you generated for your app in the Azure portal.
- **Client secret (text secret)**: Use this option to select or create a stored text secret.
- **Certificate**: Use this option to select the certificate whose public key you register (or will register) for your app in the Azure portal. Or, create a new one.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when **Ingestion mode** is set to **Streaming** and the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

> This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.



If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5 GB`. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.



Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Parquet Settings

To write out Parquet files, note that:

- On Linux, you can use the Cribl Edge CLI's `parquet` command to view a Parquet file, its metadata, or its schema.
- Cribl Edge Workers support Parquet only when running on Linux, not on Windows.
- See Parquet Schemas for pointers on how to avoid problems such as data mismatches.

Automatic schema: Toggle on to automatically generate a Parquet schema based on the events of each Parquet file that Cribl Edge writes. When toggled off (the default), exposes the following additional field:

- **Parquet schema**: Select a schema from the drop-down.



If you need to modify a schema or add a new one, follow the instructions in our Parquet Schemas topic. These steps will propagate the freshest schema back to this drop-down.

Parquet version: Determines which data types are supported and how they are represented. Defaults to `2.6`; `2.4` and `1.0` are also available.

Data page version: Serialization format for data pages. Defaults to `V2`. If your toolchain includes a Parquet reader that does not support `V2`, use `V1`.

Group row limit: The number of rows that every group will contain. The final group can contain a smaller number of rows. Defaults to `10000`.

Page size: The target memory size for page segments. Generally, lower values improve reading speed, while higher values improve compression. Must be a positive integer smaller than the **Row group size** value, with appropriate units. Defaults to `1 MB`.

Log invalid rows: Toggle on to output up to 20 unique rows that were skipped due to data format mismatch. Log level must be set to `debug` for output to be visible.

Write statistics: Leave toggled on (the default) if you have Parquet tools configured to view statistics - these profile an entire file in terms of minimum/maximum values within data, numbers of nulls, etc.

Write page indexes: Leave toggled on (the default) if your Parquet reader uses statistics from Page Indexes to enable page skipping. One Page Index contains statistics for one data page.

Write page checksum: Toggle on if you have configured Parquet tools to verify data integrity using the checksums of Parquet pages.

Metadata (optional): The metadata of files the Destination writes will include the properties you add here as key-value pairs. For example, one way to tag events as belonging to the OCSF category for security findings would be to set **Key** to `OCSF Event Class` and **Value** to `2001`.

Retries



This tab is displayed when **General Settings** > **Ingestion mode** is set to **Streaming** mode.

Honor Retry-After header: Whether to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Edge limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. When enabled, any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. When disabled, all `Retry-After` headers are ignored.

Settings for failed HTTP requests: When you want to automatically retry requests that receive particular HTTP response status codes, use these settings to list those response codes.

For any HTTP response status codes that are not explicitly configured for retries, Cribl Edge applies the following rules:

Status Code	Action
`408, 429, 503, 504, 520.`	Retry the request.
Any other status code.	Drop the request.

Upon receiving a response code that's on the list, Cribl Edge first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Edge continues retrying the request without increasing the time between retries any further.

By default, `429 (Too Many Requests)` is the only response code configured for automatic retries. For each response code you want to add to the list, click **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: When you want to automatically retry requests that have timed out, toggle this control on to display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

The controls on this tab vary depending on the **Ingestion mode** selected on the **General Settings** tab. Below you'll find one dedicated section for the Batching, and another for the Streaming controls.

Only the **Environment** control, at the bottom of the tab, is common to both:

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Advanced Settings for Batching Mode

Flush immediately: Toggle on to bypass the data management service's aggregation mechanism. See the ADX documentation.

Retain blob on success: By default, Cribl Edge deletes each data blob once ingestion is complete. Toggle this on if you prefer to retain blobs instead.

Extent tags: Optionally, add strings or tags to partitions of the target table - Azure calls these extents or data shards.

Enforce uniqueness via tag values: Use the **Add value** button to specify a list of `ingest-by` values. Cribl Edge will then check the `ingest-by` tags of incoming extents and discard those whose values match a listed value. This mechanism for avoiding ingesting duplicate extents uses the `ingestIfNotExists` property, as described in the ADX documentation.

Report level: Level for ingestion status reporting. Options are `DoNotReport`, `FailuresOnly` (the default), and `FailuresAndSuccesses`.

Report method: Target for ingestion status reporting. Options are `Queue` (the default), `Table`, and `QueueAndTable`.

Additional fields: Optionally, enter additional configuration properties to send to the ingestion service.

Staging location: Local filesystem location in which to buffer files before compressing and moving them to the final destination. Cribl recommends that this location be stable and high-performance. Defaults to `/tmp`.

File name suffix expression: The output filename suffix. Must be a JavaScript expression (which can evaluate to a constant), enclosed in quotes or backticks.

Defaults to `` `.${C.env["CRIBL_WORKER_ID"]}.${__format}${__compression === "gzip" ? ".gz" : ""}` ``, where `__format` can be `json` or `raw`, and `__compression` can be `none` or `gzip`.



To prevent files from being overwritten, Cribl appends a random sequence of 6 characters to the end of their names. This also applies to prefix and suffix expressions in file names.

For example, if you set the **File name prefix expression** to `CriblExec` and set the **File name suffix expression** to `.csv`, the file name might display as `CriblExec-adPRWM.csv` with `adPRWM` appended.

File size limit (MB): Maximum uncompressed output file size. Files reaching this size will be closed and moved to the storage container. Defaults to `32`.

File open time limit (sec): Maximum amount of time to write to a file. Files open for longer than this limit will be closed and moved to the storage container. Defaults to `300` seconds (5 minutes).

Idle time limit (sec): Maximum amount of time to keep inactive files open. Files open for longer than this limit will be closed and moved to the storage container. Default: `30`.

Open file limit: Maximum number of files to keep open concurrently. When exceeded, the oldest open files will be closed and moved to the storage container. Default: `100`.

Concurrent file parts limit: Maximum number of parts to upload in parallel per file. A value of `1` tells the Destination to send one part at a time - that is, to upload the file's contents sequentially. Defaults to `1`; highest allowed value is `10`.

Disk space protection: Specifies whether to `Block` (default) or `Drop` incoming events when the disk space falls below the globally defined Min free disk space amount.

Add output ID: Toggle on if you want Cribl Edge to append the Destination name to staging directory pathnames. This can make it easier to organize and troubleshoot when you have multiple Destinations populating staging directories.

Remove empty staging directories: When toggled on (the default), Cribl Edge deletes empty staging directories after moving files. This prevents the proliferation of orphaned empty directories. When enabled, exposes this additional option:

- **Staging cleanup period**: How often (in seconds) to delete empty directories when **Remove staging dirs** is enabled. Defaults to `300` seconds (every 5 minutes). Minimum configurable interval is `10` seconds; maximum is `86400` seconds (every 24 hours).

- **Directory batch size**: Specifies how many directories are scanned per batch during automatic cleanup of empty staging directories. Set between `10` and `10000`, defaults to `1000`. Higher values speed up cleanup but increase memory usage.

Enable dead-lettering: Toggle this on to set a maximum number of retries, and to move files to a designated directory when write failures exceed that limit. This prevents data flow blockage and excessive error logging due to undeliverable files. When enabled, exposes two additional fields:

- **Dead-letter location**: Specify the storage location for undeliverable files. Defaults to `$CRIBL_HOME/state/outputs/dead-letter`.
- **Maximum retry limit**: Configure the retry limit for failed file deliveries. This setting defines how many times the system will attempt to move a file to its intended location before it is deemed undeliverable and placed in the dead-letter directory. Defaults to `20`.

Advanced Settings for Streaming Mode

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests per Worker Process. When Cribl Edge hits this limit, it begins throttling traffic to the downstream service. Defaults to `5`. Minimum: `1`. Maximum: `32`.

Body size limit (KB): Maximum size of the request body before compression. Defaults to `4096` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Body size limit** value until downstream receivers reliably accept all events.

Events-per-request limit: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low settings could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Validate server certs: When toggled on (default) Cribl Edge will reject certificates that are **not** authorized by a CA in the **CA certificate path**, nor by another trusted CA (such as the system's CA, for example).

Round-robin DNS: Toggle on to enable round-robin DNS lookup across multiple IP addresses, IPv4 and IPv6. When a DNS server resolves a Fully Qualified Domain Name (FQDN) to multiple IP addresses, Cribl Edge will sequentially use each address in the order they are returned by the DNS server for subsequent connection attempts.

Keep alive: By default, Cribl Edge sends `Keep-Alive` headers to the remote server and preserves the connection from the client side up to a maximum of 120 seconds. Toggle this off if you want Cribl Edge to close the connection immediately after sending a request.

Azure Permissions

When configuring an Azure Destination in Cribl Edge, you must grant the necessary permissions for it to interact with your Azure environment. Cribl recommends creating an Azure service principal with the minimum required permissions using Azure role-based access control (Azure RBAC). This approach ensures Cribl Edge has a dedicated identity with precisely the access it needs, following the principle of least privilege.

The minimum permissions required for Destinations are:

- **Database Ingestor:** For the database being written to.
- **Database Viewer:** Required if **Validate database settings** is enabled.

OR

- **Table Ingestor:** For the table being written to.
- **Database Viewer:** Required if **Validate database settings** is enabled.

See Manage Azure Data Explorer cluster permissions and Manage Azure Data Explorer database permissions in the Azure portal in the official Microsoft documentation for more information.

Internal Fields

Cribl Edge uses a set of internal fields to assist in forwarding data to a Destination.

Field for this Destination:

- `__partition`

Proxying Requests

If you need to proxy HTTP/S requests, see System Proxy Configuration.

Resources

For examples of configuring Cribl Edge to interoperate with Azure services, see these guides:

- Preparing the Azure Workspace.
- SSO with Microsoft Entra ID (Cribl.Cloud).
- SSO with Microsoft Entra ID (On-Prem).

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

12.3.4.3. Azure Event Hubs Destination

Cribl Edge supports sending data to Azure Event Hubs.

Type: Streaming | TLS Support: Configurable | PQ Support: Yes

Azure Event Hubs uses a binary protocol over TCP. It does not support HTTP proxies, so Cribl Edge must send events directly to receivers. You might need to adjust your firewall rules to allow this traffic.

For configuration examples, see Resources below.

See Microsoft's Compare Azure Event Hubs Tiers topic to configure tiers whose features and quotas match your desired data volume and throughput.

Configuring Cribl Edge to Output to Azure Event Hubs

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this Azure Event Hubs definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
    - **Description**: Optionally, enter a description.
    - **Brokers**: List of Event Hub Kafka brokers to connect to. (For example, `yourdomain.servicebus.windows.net:9093`.) Find the hostname in Shared Access Policies, in the host portion of the primary or secondary connection string. If omitted, the port will default to `9093`.
    - **Event Hub name**: The name of the Event Hub (a.k.a., Kafka Topic) on which to publish events. Can be overwritten using the `__topicOut` field.
3. Next, you can configure the following **Optional Settings**:

- **Acknowledgments**: Control the number of required acknowledgments. Defaults to `Leader`. Setting this control to `Leader` or `All` may significantly slow throughput. If the impact is too great for your use case, set this control to `None`.
- **Record data format**: Format to use to serialize events before writing to the Event Hub Kafka brokers. Defaults to `JSON`.
  - `JSON` serializes the entire event from your Pipeline, including all metadata and enriched fields, into a single JSON object.
  - `Field _raw` sends only the contents of the `_raw` field as the payload. This option provides maximum throughput and minimal overhead.
- **Backpressure behavior**: Whether to block, drop, or queue events when all receivers are exerting backpressure. Defaults to `Block`.
- **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

4. Optionally, you can adjust the TLS, Authentication, Persistent Queue, Processing, Parquet, Retries and Advanced settings outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

TLS Settings (Client Side)

Enabled: Defaults to toggled on.

Validate server certs: Defaults to toggled on.

Authentication

Authentication parameters to use when connecting to brokers. Using TLS is highly recommended.

Enabled: If toggled on (default), this section's remaining settings are displayed, and all are required settings.

SASL mechanism: SASL (Simple Authentication and Security Layer) authentication mechanism to use with Kafka brokers. Defaults to `PLAIN`, which exposes Basic Authentication options that rely on Azure Event Hubs connection strings. Select `OAUTHBEARER` to enable OAuth Authentication via a different set of options.

Basic Authentication

Selecting the `PLAIN` SASL mechanism provides the options listed in this section.

Username: The username for authentication. For Event Hubs, this should always be `$ConnectionString`.

Authentication method: Use the buttons to select one of these options:

- **Manual**: Use this default option to enter your Event Hubs connection string's primary or secondary key from the Event Hubs workspace. Exposes a **Password** field for this purpose.
- **Secret**: This option exposes a **Password (text secret)** drop-down, in which you can select a stored secret that references an Event Hubs connection string. The secret can reside in Cribl Edge's internal secrets manager or (if enabled) in an external KMS. A **Create** link is available if you need a new secret.

Connection String Format

Either authentication method above uses an Azure Event Hubs connection string in this format:

```
Endpoint=sb://<FQDN>/;SharedAccessKeyName=<your-shared-access-key-name>;SharedAccessKey=<your-shared-access-key-value>
```

A fictitious example is:

```
Endpoint=sb://dummynamespace.servicebus.windows.net/;SharedAccessKeyName=DummyAccessKeyN
```

OAuth Authentication

Selecting the `OAUTHBEARER` SASL mechanism provides the options listed in this section.

Microsoft Entra ID authentication endpoint: Specifies the Microsoft Entra ID endpoint from which to acquire authentication tokens. Defaults to `https://login.microsoftonline.com`. You can instead select `https://login.microsoftonline.us` or `https://login.partner.microsoftonline.cn`.

Client ID: Enter the `client_id` to pass in the OAuth request parameter.

Tenant identifier: Enter your Microsoft Entra ID subscription's directory ID (tenant ID).

Scope: Enter the scope to pass in the OAuth request parameter. This will be of the form: `https://<Event-Hubs-Namespace-Host-name>/.default`. (For example, for an Event Hubs Namespace > Host name: `goatyoga.servicebus.windows.net`, **Scope**: `https://goatyoga.servicebus.windows.net/.default`.)

Authentication method: Use the buttons to select one of these options:

- **Manual**: This default option exposes a **Client secret** field, in which to directly enter the `client_secret` to pass in the OAuth request parameter.
- **Secret**: Exposes a **Client secret (text secret)** drop-down, in which you can select a stored secret that references the `client_secret`. A **Create** link is available to define a new secret.
- **Certificate**: Exposes a **Certificate name** drop-down, in which you can select a stored certificate. A **Create** link is available to define a new cert.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.



If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5` GB. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.



Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Retries

These settings provide flexibility in handling retries for failed messages, allowing you to balance between quick retries and avoiding excessive load on the system.

The default configuration starts with a 300 milliseconds retry interval, doubles the interval after each retry, and caps the maximum retry interval at 30 seconds. The system will attempt to retry the request up to five times before considering it a failure.

Retry limit: Maximum number of times to retry a failed request before the message fails. Defaults to `5`. Enter `0` to not retry at all. For example, if set to `5`, the system will attempt to retry the request up to five times before considering it a failure.

Initial retry interval (ms): Initial value used to calculate the retry interval, in milliseconds. This value determines the starting point for the retry delay. The default (and minimum) is `300` ms (three seconds). The maximum allowed value is `600000` ms (10 minutes). For example, if set to `1000` ms, the first retry will occur after one second.

Backoff multiplier: Set the backoff multiplier (`2`–`20`) to control the retry frequency for failed messages. The multiplier is used in an exponential backoff formula to increase the delay between retries. For faster retries, use a lower multiplier. For slower retries with more delay between attempts, use a higher multiplier. For example, with an initial retry interval of `1000` ms and a multiplier of `2`, the retry intervals will be 1,000 ms, 2,000 ms, 4,000 ms, and so on. See the Kafka documentation for details.

Backoff limit (ms): The maximum wait time for a retry, in milliseconds. This setting caps the exponential backoff delay to prevent excessively long wait times. The default (and minimum) value is `30000` ms (30

seconds), and the maximum is `180000` ms (180 seconds). For example, if the calculated retry interval exceeds 180,000 ms, the retry will occur after 180,000 ms instead.

Advanced Settings

Record size limit (KB, uncompressed): Maximum size (KB) of each record batch before compression. Setting should be < `message.max.bytes` settings in Kafka brokers. Defaults to `768`.

Events-per-batch limit: Maximum number of events in a batch before forcing a flush. Defaults to `1000`.

Flush period (sec): Maximum time between requests. Low settings could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Connection timeout (ms): Maximum time to wait for a successful connection. Defaults to `10000` ms (10 seconds). Valid range is `1000` to `3600000` ms (1 second to 1 hour).

Request timeout (ms): Maximum time to wait for a successful request. Defaults to `60000` ms (1 minute).

Authentication timeout (ms): Maximum time to wait for Kafka to respond to an authentication request. Defaults to `1000` (1 second).

Reauthentication threshold (ms): If the broker requires periodic reauthentication, this setting defines how long before the reauthentication timeout Cribl Edge initiates the reauthentication. Defaults to `10000` (10 seconds).

A small value for this setting, combined with high network latency, might prevent the Destination from reauthenticating before the Kafka broker closes the connection.

A large value might cause the Destination to send reauthentication messages too soon, wasting bandwidth.

The Kafka setting `connections.max.reauth.ms` controls the reuthentication threshold on the Kafka side.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Working with Event Timestamps

By default, when an incoming event reaches this Destination, the underlying Kafka JS library adds a `timestamp` field set to the current time at that moment. The library gets the current time from the `Date.now()` JavaScript method. Events that this Destination sends to downstream Kafka receivers include this `timestamp` field.

Some production situations require the `timestamp` value to be the time an incoming event was originally created by an upstream sender, not the current time of its arrival at this Destination. For example, suppose the events were originally created over a wide time range, and arrive at the Destination hours or days later. In this

case, the original creation time might be important to retain in events Cribl Edge sends to downstream Kafka receivers.

Here is how to satisfy such a requirement:

1. In a Pipeline that routes events to this Destination, use an Eval Function to add a field named `__kafkaTime` and write the incoming event's original creation time into that field. The value of `__kafkaTime` **must** be in Unix time (either seconds or milliseconds - both will work). For context, see this explanation of the related fields `_time` and `__origTime`.
2. This Destination will recognize the `__kafkaTime` field and write its value into the `timestamp` field. (This is similar to the way you can use the `__topicOut` field to overwrite a topic setting, as described above.)

If the `__kafkaTime` field is **not** present, the Destination will apply the default behavior (using `Date.now()`) described previously.

Azure Permissions

When configuring an Azure Destination in Cribl Edge, you must grant the necessary permissions for it to interact with your Azure environment. Cribl recommends creating an Azure service principal with the minimum required permissions using Azure role-based access control (Azure RBAC). This approach ensures Cribl Edge has a dedicated identity with precisely the access it needs, following the principle of least privilege.

The minimum permissions required for Destinations are:

- **Azure Event Hubs Data Sender**

See Authorize access to Azure Event Hubs resources using Microsoft Entra ID in the official Microsoft documentation for more information.

Internal Fields

Cribl Edge uses a set of internal fields to assist in forwarding data to a Destination.

Fields for this Destination:

- `__topicOut`
- `__key`
- `__headers`
- `__keySchemaIdOut`
- `__valueSchemaIdOut`

Resources

For examples of configuring Cribl Edge to interoperate with Azure services, see these guides:

- Preparing the Azure Workspace.
- SSO with Microsoft Entra ID (Cribl.Cloud).
- SSO with Microsoft Entra ID (On-Prem).

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

Cribl University offers an Advanced Troubleshooting > Destination Integrations: Azure Event Hubs short course. To follow the direct course link, first log into your Cribl University account. (To create an account, select the **Sign up** link. You'll need to click through a short **Terms & Conditions** presentation,

12.3.4.4. Azure Monitor Logs Destination

The Azure Monitor Log Destination is now deprecated. The replacement for routing data to the Microsoft Log Analytics platform is the Microsoft Sentinel Destination. This change aligns with the Microsoft recommendation to use the Azure Monitor Logs Ingestion API, which the Microsoft Sentinel Destination is built upon.

Microsoft is retiring the legacy HTTP Data Collector API on September 14, 2026. For details, see the Microsoft deprecation notice: Send log data to Azure Monitor by using the HTTP Data Collector API (deprecated).

Cribl Edge supports sending data to Azure Monitor Logs.

Type: Streaming | TLS Support: Yes | PQ Support: Yes

For configuration examples, see Resources below.

Configure Cribl Edge to Output to Azure Monitor Logs

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this Azure Monitor Logs definition.
    - **Description**: Optionally, enter a description.
    - **Log type**: The Record Type of events sent to this LogAnalytics workspace. Defaults to `Cribl`. Use only letters, numbers, and `_` characters. (See Microsoft's Azure Monitor documentation.) Can be overwritten by an event's `__logType` field.
3. Under **Authentication**, select an **Authentication method** from the dropdown:

- **Manual**: Displays fields in which to enter your Azure Log Analytics **Workspace ID** and your Primary or Secondary Shared **Workspace key**. See the Azure Monitor documentation.
- **Secret**: This option exposes a **Secret key pair** drop-down, in which you can select a stored secret that references the credentials described above. A **Create** link is available to store a new, reusable secret.

4. Next, you can configure the following **Optional Settings**:
   - **DNS name of API endpoint**: Enter the DNS name of the Log API endpoint that sends log data to a Log Analytics workspace in Azure Monitor. Defaults to: `.ods.opinsights.azure.com.` Cribl Edge will add a prefix and suffix around this DNS name, to construct a URI in this format: `https://<Workspace_ID><your_DNS_name>/api/logs?api-version=<API version>`.
   - **Resource ID**: Resource ID of the Azure resource to associate the data with. This populates the `_ResourceId` property, and allows the data to be included in resource-centric queries. (Optional, but if this field is not specified, the data will not be included in resource-centric queries.)
   - **Backpressure behavior**: Whether to block, drop, or queue events when all receivers are exerting backpressure. Defaults to `Block`.
   - **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
5. Optionally, you can adjust the Persistent Queue, Processing, Retries and Advanced settings outlined in the sections below.
6. Select **Save**, then **Commit & Deploy**.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

> This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

> If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5 GB`. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.



Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Retries

Honor Retry-After header: Toggle on to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream/Edge limits the delay to 180 seconds even if the `Retry-After`

header specifies a longer delay. Any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. Toggle off to ignore all `Retry-After` headers.

Default Retry Behavior: By default, Cribl Stream/Edge applies predefined retry strategies for certain failed requests to the Destination using an exponential backoff algorithm. You can override these settings by manually configuring the retry behavior for a particular error code.



Destinations created before the 4.16.0 release do not support the default `4xx` series retry behavior. If your Destination was created prior to this release and you'd like it to support the behavior, reconfigure the Destination.

To retry requests, Cribl Stream/Edge applies the following rules by default:

Status Code	Action
Any in the `1xx`, `3xx` series	Drop the request
`401`, `403`, `408`, `429` errors	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached
Any other `4xx` series	Drop the request
Any in the `5xx` series	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached

By default, any `429 (Too Many Requests)` response that is not listed in the retry settings is treated as a non-retryable `4xx` error and dropped. If you want `429` responses retried, ensure `429` is included in the **Settings for failed HTTP requests**.

Settings for failed HTTP requests: You can also manually configure Cribl Stream/Edge to automatically retry requests that receive particular HTTP response status codes.

Upon receiving a response code that's on the list, Cribl Stream/Edge first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream/Edge continues retrying the request without increasing the time between retries any further.

If the sender (which manages the connection to the Destination) is at capacity, it will not accept any incoming events. These incoming events originate internally from a previous stage of the data flow when Destinations send outbound requests to their respective external services, and they include retry requests and new requests. Any events that were already in transit when the sender reached capacity will continue to be processed downstream.

Sender capacity is freed up when an outgoing request succeeds or encounters a non-retryable error. When the sender has available capacity again, it will resume accepting incoming events. This capacity management is influenced by the number of active connections and configured limits, such as concurrency and buffer sizes. If a Pipeline sends events faster than the Destination can process, the buffers may fill up, leading to backpressure and `Sender at capacity` warnings. This backpressure prevents the sender from accepting additional requests until capacity is restored.

For each response code you want to add to the list, select **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: Toggle on to automatically retry requests that have timed out and display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Toggle on to reject certificates that are **not** authorized by a CA in the **CA certificate path**, nor by another trusted CA (for example, the system's CA).

Round-robin DNS: Toggle on to enable round-robin DNS lookup across multiple IP addresses, IPv4 and IPv6. When a DNS server resolves a Fully Qualified Domain Name (FQDN) to multiple IP addresses, Cribl Edge will sequentially use each address in the order they are returned by the DNS server for subsequent connection attempts.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests per Worker Process. When Cribl Edge hits this limit, it begins throttling traffic to the downstream service. Defaults to `5`. Minimum: `1`. Maximum: `32`.

Body size limit (KB): Maximum size of the request body before compression. Defaults to `1024` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Body size limit** value until downstream receivers reliably accept all events.

Events-per-request limit: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low settings could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Extra HTTP headers: Name-value pairs to pass as additional HTTP headers.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Cribl Edge will redact all headers, except non-sensitive headers that you declare below in **Safe headers**.

Safe headers: Add headers to declare them as safe to log in plaintext. (Sensitive headers such as `authorization` will always be redacted, even if listed here.) Use a tab or hard return to separate header names.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Azure Monitor Limitations

The Azure Monitor Logs architecture limits the number of columns per table, characters per column name, and other parameters. For details, see Microsoft's Azure Monitor Service Limits topic.

Azure will drop logs if your data exceeds these limits. To diagnose this, you can search in the Azure Data Explorer console with a query like this:

```
Operation | summarize count() by Detail
```

...for error messages of this form:

```
Data of type <type> was dropped: The number of custom fields <number> is above the limit of 500 fields per data type.
```

Notes on HTTP-Based Outputs

- To proxy outbound HTTP/S requests, see System Proxy Configuration.
- Cribl Edge will attempt to use keepalives to reuse a connection for multiple requests. After two minutes of the first use, the connection will be thrown away, and a new one will be reattempted. This is to prevent sticking to a particular Destination when there is a constant flow of events.
- If keepalives are not supported by the server (or if the server closes a pooled connection while idle), a new connection will be established for the next request.
- When resolving the Destination's hostname, Cribl Edge will pick the first IP in the list for use in the next connection. Enable Round-robin DNS to better balance distribution of events between destination cluster nodes.

Azure Permissions

Azure Monitor Logs does not require any specific permissions.

Resources

For examples of configuring Cribl Edge to interoperate with Azure services, see these guides:

- Preparing the Azure Workspace.
- SSO with Microsoft Entra ID (Cribl.Cloud).
- SSO with Microsoft Entra ID (On-Prem).

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

12.3.4.5. Microsoft Sentinel Destination

Cribl Edge can send log and metric events to the Microsoft Sentinel SIEM. This Destination encrypts and sends events via the HTTPS protocol. The supported environments include:

- Azure Public Cloud
- Azure US Government Cloud
- Microsoft Azure operated by 21Vianet



Type: Streaming | TLS Support: No | PQ Support: Yes

For a walk-through of configuring this Destination with Microsoft Sentinel, see our Microsoft Sentinel SIEM Integration guide. Also see these Packs on the Cribl Packs Dispensary, which provide processing Pipelines that you can import and adapt to your needs:

- Microsoft Sentinel Pack.
- Microsoft Sentinel Syslog Pack.
- Microsoft Sentinel Common Security Log Pack.



Ensure that the field names in your data match the schema defined in the Data Collection Rule (DCR) for Microsoft Sentinel. The DCR specifies the expected structure, including field names and types. Mismatched field names can cause dropped events. This Destination sends out individual top-level fields, not a combined `_raw` (or similar) field.

Prerequisites

Before configuring your Microsoft Sentinel Destination(s), you have to prepare the Azure workspace, create data collection rules (DCRs), and obtain the ingestion URL that will receive the data from your Destination(s).

Complete these preparatory steps, which are described in Microsoft Sentinel Integration, before configuring the Microsoft Sentinel Destination. That topic explains the overall Sentinel/Cribl workflow, provides necessary context, and shows you how to obtain the values you'll need to configure the Destination.

If you intend to send multiple event types, or tables, to Sentinel, you'll need a separate Destination for each event type. There are three approaches to configure this:

- **Multiple Destinations**: Configure separate Microsoft Sentinel Destinations, each with its own URL, and use Routes to direct data to the appropriate Destination.

- **Multiple Destinations and a Output Router**: Use an Output Router to split events from a single Source into multiple Destinations, each corresponding to a specific table.
- **Single Destination and overwriting the URL in a Pipeline**: Configure a single Microsoft Sentinel Destination and dynamically overwrite the URL within Pipelines using the `__url` internal field to direct events to the correct table.



Behind the scenes, this Destination uses the Azure Monitor Logs Ingestion API.

Configure Cribl Edge to Output to Microsoft Sentinel

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this HTTP output definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
    - **Description**: Optionally, enter a description.
    - **Endpoint configuration**: Use the buttons to select **URL** or **ID**. This specifies the method for configuring the endpoint. For details, see Endpoint Configuration Options
3. Next, you can configure the following **Optional Settings**:
    - **Backpressure behavior**: Whether to block, drop, or queue events when all receivers are exerting backpressure.
    - **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
4. Optionally, you can adjust the Authentication Settings, Processing, Retries and Advanced settings outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

Endpoint Configuration Options

For Syslog, Cribl recommends starting from the Syslog DCR template in the Cribl-Microsoft GitHub repository. This template defines a stream name of `Custom-Syslog` and routes data into the Sentinel Syslog table.

After you create a data collection rule (DCR) from this template in Azure, you will use its values when configuring the Microsoft Sentinel Destination.

Note that the Syslog template is provided under the `DataCollectionRules` path and is intended for the **DCR kind:direct** approach, matching the corresponding method described in preparing the Azure workspace for Cribl integrations. If your deployment instead uses a Data Collection Endpoint (DCE), use the DCE-based template from the same repository and follow the DCE path in the guide.

The **Endpoint configuration** setting offers the following options:

- **URL** - lets you directly enter the data collection endpoint. This method is the simplest way of configuring the endpoint. See Obtaining a URL for more information.
- **ID** - lets you enter individual IDs that Cribl Edge uses to create the URL used as the data collection endpoint.

Selecting `URL` exposes the following field.

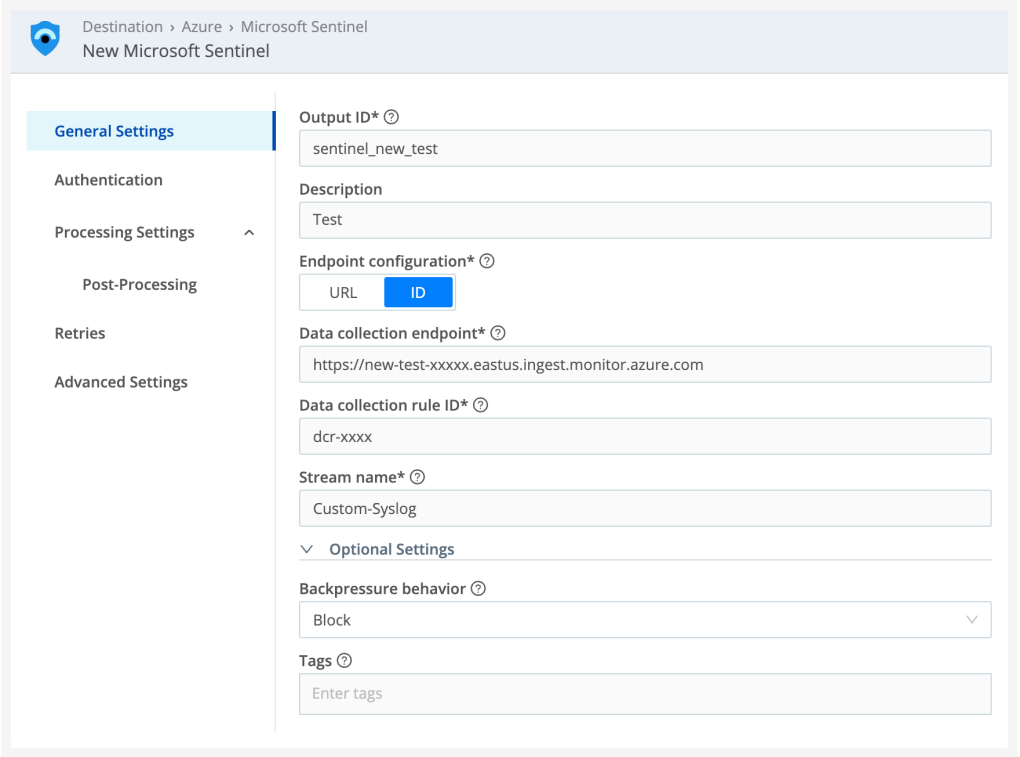
URL: Endpoint URL to send events to. The internal field `__url`, where present in events, will override the `URL` and `ID` values. See Internal Fields below.

Selecting `ID` exposes the following fields.

Data collection endpoint: The Azure `LogsIngestion` URL in the format `https://<endpoint-name>.<identifier>.<region>.ingest.monitor.azure.com`.

Data collection rule ID: Immutable ID for the DCR.

Stream name: Name of the Sentinel table in which to store events. For DCR-based ingestion, this value must exactly match the stream/table name defined in the Azure DCR template, not the friendly table name shown in the Sentinel UI. For the Cribl Syslog template, the correct stream/table name is `Custom-Syslog`. If Stream name does not exactly match the table name in the template, data may fail to land in the expected Sentinel table.



Microsoft Sentinel Destination settings

Optional Settings

Authentication Settings

In Cribl Stream, open your **Microsoft Sentinel Destination** and select **Authentication Settings**. Fill in the fields as follows:

Login URL: Endpoint for the OAuth API call. Use the OAuth2 token endpoint that matches your cloud shown in the Azure **Endpoints** view for your Microsoft Entra app.

Typical formats are:

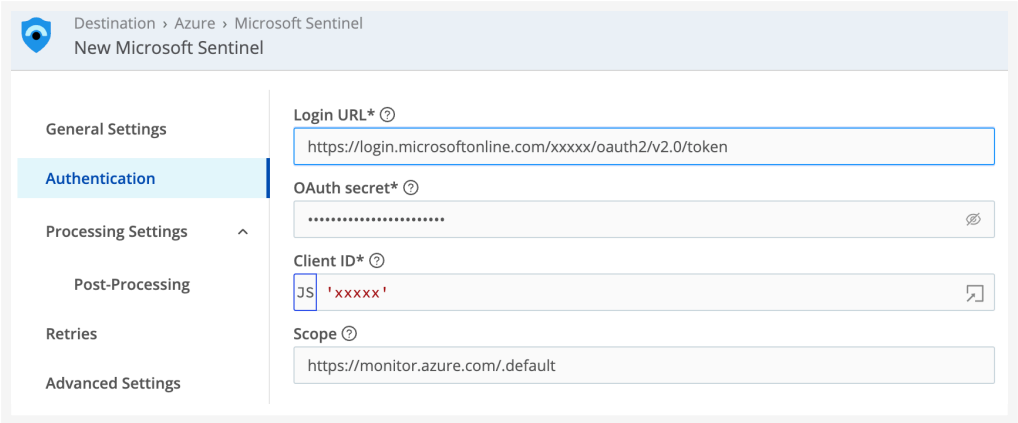
- Azure Public Cloud: `https://login.microsoftonline.com/<tenant-id>/oauth2/v2.0/token`
- Azure US Government Cloud: `https://login.microsoftonline.us/<tenant-id>/oauth2/v2.0/token`
- Microsoft Azure operated by 21Vianet: `https://login.chinacloudapi.cn/<tenant-id>/oauth2/v2.0/token`

OAuth secret: Secret parameter value to pass in the API call's request body. Secret parameter value to pass in the API call's request body. Use the client secret value you created for the app.

Client ID: JavaScript expression used to compute the application (client) ID for the Azure application. Can be a constant.

Scope: The OAuth scope to the appropriate endpoint URL for your Azure environment. This ensures that your OAuth tokens are valid and that your data is correctly ingested into Microsoft Sentinel. This setting should match the specific endpoint for your Azure environment:

- Azure Public Cloud (default): `https://monitor.azure.com/.default`
- Azure US Government Cloud: `https://monitor.azure.us/.default`
- Microsoft Azure operated by 21Vianet: `https://monitor.azure.cn/.default`



Microsoft Sentinel Destination auth settings

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

> On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.
>
> This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.



If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5 GB`. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.



Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Retries

Honor Retry-After header: Toggle on to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream/Edge limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. Any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. Toggle off to ignore all `Retry-After` headers.

Default Retry Behavior: By default, Cribl Stream/Edge applies predefined retry strategies for certain failed requests to the Destination using an exponential backoff algorithm. You can override these settings by

manually configuring the retry behavior for a particular error code.

To retry requests, Cribl Stream/Edge applies the following rules by default:

Status Code	Action
Any in the `1xx`, `3xx` series	Drop the request
`408`, `429` errors	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached
Any other `4xx` series	Drop the request
Any `5xx` series	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached

By default, any `429 (Too Many Requests)` response that is not listed in the retry settings is treated as a non-retryable `4xx` error and dropped. If you want `429` responses retried, ensure `429` is included in the **Settings for failed HTTP requests**.

Settings for failed HTTP requests: You can also manually configure Cribl Stream/Edge to automatically retry requests that receive particular HTTP response status codes.

Upon receiving a response code that's on the list, Cribl Stream/Edge first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream/Edge continues retrying the request without increasing the time between retries any further.

If the sender (which manages the connection to the Destination) is at capacity, it will not accept any incoming events. These incoming events originate internally from a previous stage of the data flow when Destinations send outbound requests to their respective external services, and they include retry requests and new requests. Any events that were already in transit when the sender reached capacity will continue to be processed downstream.

Sender capacity is freed up when an outgoing request succeeds or encounters a non-retryable error. When the sender has available capacity again, it will resume accepting incoming events. This capacity management is influenced by the number of active connections and configured limits, such as concurrency and buffer sizes. If a Pipeline sends events faster than the Destination can process, the buffers may fill up, leading to backpressure and `Sender at capacity` warnings. This backpressure prevents the sender from accepting additional requests until capacity is restored.

For each response code you want to add to the list, select **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: Toggle on to automatically retry requests that have timed out and display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Reject certificates that are not authorized by a trusted CA (for example, the system's CA). Toggle off if you want Cribl Edge to accept such certificates. Defaults to toggled on.

Round-robin DNS: Toggle on to enable round-robin DNS lookup across multiple IP addresses, IPv4 and IPv6. When a DNS server resolves a Fully Qualified Domain Name (FQDN) to multiple IP addresses, Cribl Edge will sequentially use each address in the order they are returned by the DNS server for subsequent connection attempts.

By default, Cribl Edge enables gzip-compress to compress the payload body before sending. Toggle this off if you want Cribl Edge not to gzip-compress the payload body.

Keep alive: By default, Cribl Edge sends `Keep-Alive` headers to the remote server and preserves the connection from the client side up to a maximum of 120 seconds. Toggle this off if you want Cribl Edge to close the connection immediately after sending a request.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests per Worker Process. When Cribl Edge hits this limit, it begins throttling traffic to the downstream service. Defaults to `5`. Minimum: `1`. Maximum: `32`.

Body size limit (KB): Maximum size of the request body before compression. Defaults to `1000` KB, the maximum allowed size of API call. Be aware that high values can cause high memory usage per Worker Node, especially if a dynamically constructed URL (see Internal Fields) causes this Destination to send events to more than one URL. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Body size limit** value until downstream receivers reliably accept all events.

Buffer memory limit (KB): Total amount of memory used to buffer outgoing requests waiting to be sent. If left blank, defaults to 5 times the max body size (if set). If 0, no limit is enforced. This provides granular control over the memory allocated for buffering outgoing batched requests. Increasing the limit allows batches to grow larger before being flushed, improving efficiency for data with high cardinality (a large number of unique batches). Finding the optimal balance between efficient data transfer and memory usage involves adjusting both the **Buffer memory limit** and **Body size limit** settings.

Events-per-request limit: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Extra HTTP headers: Name-value pairs to pass to all events as additional HTTP headers. Values will be sent encrypted. You can also add headers dynamically on a per-event basis in the `__headers` field. See Internal Fields below.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Cribl Edge will redact all headers, except non-sensitive headers that you declare below in **Safe headers**.

Safe headers: Add headers to declare them as safe to log in plaintext. (Sensitive headers such as `authorization` will always be redacted, even if listed here.) Use a tab or hard return to separate header names.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Internal Fields

Cribl Edge uses a set of internal fields to assist in forwarding data to a Destination.

Fields for this Destination:

- `__criblMetrics`
- `__url`
- `__headers`

If an event contains the internal field `__criblMetrics`, Cribl Edge will send it to the HTTP endpoint as a metric event. Otherwise, Cribl Edge will send it as a log event.

If an event contains the internal field `__url`, that field's value will override the **General Settings** > **URL** value. This way, you can determine the endpoint URL dynamically.

For example, you could create a Pipeline containing an Eval Function that adds the `__url` field, and connect that Pipeline to your Microsoft Sentinel Destination. Configure the Eval Function to set `__url`'s value to a URL that varies depending on a global variable, or on some property of the event, or on some other dynamically-generated value that meets your needs.

If an event contains the internal field `__headers`, that field's value will be a JSON object containing Name-value pairs, each of which defines a header. By creating Pipelines that set the value of `__headers` according to conditions that you specify, you can add headers dynamically.

For example, you could create a Pipeline containing Eval Functions that add the `__headers` field, and connect that Pipeline to your Microsoft Sentinel Destination. Configure the Eval Functions to set `__headers` values to Name-value pairs that vary depending on some properties of the event, or on dynamically-generated values that meet your needs.

Here's an overview of how to add headers:

- To add "dynamic" (a.k.a. "custom") headers to some events but not others, use the `__headers` field.
- To define headers to add to **all** events, use **Advanced Settings** > **Extra HTTP Headers**.
- An event can include headers added by both methods. So if you use `__headers` to add `{ "api-key": "foo" }` and **Extra HTTP Headers** to add `{ "goat": "Kid A" }`, you'll get both headers.
- Headers added via the `__headers` field take precedence. So if you use `__headers` to add `{ "api-key": "foo" }` and **Extra HTTP Headers** to add `{ "api-key": "bar" }`, you'll get only one header, and that will be `{ "api-key": "foo" }`.

Notes on HTTP-Based Outputs

- To proxy outbound HTTP/S requests, see System Proxy Configuration.
- Cribl Edge will attempt to use keepalives to reuse a connection for multiple requests. After two minutes of the first use, it will throw away the connection and attempt a new one. This is to prevent sticking to a particular destination when there is a constant flow of events.
- If the server does not support keepalives (or if the server closes a pooled connection while idle), Cribl Edge will establish a new connection for the next request.
- When resolving the Destination's hostname, Cribl Edge will pick the first IP in the list for use in the next connection. Enable **Round-robin DNS** to better balance distribution of events among destination cluster nodes.

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

12.3.5. Elastic

You can send data from Cribl Edge to the following Elastic services:

Destination	Description
Elasticsearch	Send events to an Elasticsearch cluster using the Bulk API
Elastic Cloud	Send events to Elastic Cloud

12.3.5.1. Elasticsearch Destination

Cribl Edge can send events to an Elasticsearch cluster using the Bulk API. As of v.3.3, Cribl Edge supports Elastic data streams.

Use the Elastic Cloud Destination if you are exclusively targeting Elastic Cloud and prefer a simpler, optimized configuration process.



Type: Streaming | TLS Support: Configurable | PQ Support: Yes

For an example of using Cribl Edge to parse and enrich inbound events for Elasticsearch, see our Splunk to Elasticsearch topic.

Configure Cribl Edge to Output to Elasticsearch

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this Elasticsearch Destination definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
    - **Description**: Optionally, enter a description.
    - **Load balancing**: Toggle on (default) to specify multiple bulk API URLs and load weights. If toggled off and you notice that Cribl Edge is not sending data to all possible IP addresses, enable **Advanced Settings** > **Round-robin DNS**.
    - **Bulk API URL or Cloud ID**: Specify either an Elasticsearch cluster or Elastic Cloud deployment to send events to. For an Elasticsearch cluster, enter a URL (for example, `http://<myElasticCluster>:9200/_bulk`). For an Elastic Cloud deployment, enter its Cloud ID. This setting is not available when **Load balancing** is enabled.
    - **Index or data stream**: Enter a JavaScript expression that evaluates to the name of the Elastic data stream or Elastic index where you want events to go. The expression is evaluated for each event,

can evaluate to a constant value, and must be enclosed in quotes or backticks. An event's `__index` field can overwrite the index or data stream name.

3. Under **Authentication**, toggle **Authentication enabled** on to display **Authentication method** dropdown. Select one of these options:
    - **Manual**: Enter your credentials directly in the resulting **Username** and **Password** fields.
    - **Secret**: Exposes a **Credentials secret** drop-down, in which you can select a stored secret that references the credentials described above. A **Create** link is available to store a new, reusable secret.
    - **Manual API Key**: Exposes an **API key** field to directly enter your Elasticsearch API key.
    - **Secret API Key**: Exposes an **API key (text secret)** drop-down, in which you can select a stored text secret that references your Elasticsearch API key. A **Create** link is available to store a new, reusable secret.
4. Next, you can configure the following **Optional Settings**:
    - **Exclude current host IPs**: This toggle appears when **Load balancing** is toggled on. It determines whether to exclude all IPs of the current host from the list of any resolved hostnames. Default is toggled off, which keeps the current host available for load balancing.
    - **Type**: Specify document type to use for events. An event's `__type` field can overwrite this value.
    - **Backpressure behavior**: Specify whether to block, drop, or queue events when all receivers are exerting backpressure. Defaults to `Block`.
    - **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
5. Optionally, you can adjust the Persistent Queue, Processing, Retries, and Advanced settings outlined in the sections below.
6. Select **Save**, then **Commit & Deploy**.

Bulk API URLs

Use the **Bulk API URLs** table to specify a known set of receivers on which to load-balance data. To specify more receivers on new rows, click **Add URL**. Each row provides the following fields:

URL: Specify the URL to an Elastic node to send events to - for example, `http://elastic:9200/_bulk`

Load weight: Set the relative traffic-handling capability for each connection by assigning a weight (> `0`). This column accepts arbitrary values, but for best results, assign weights in the same order of magnitude to all connections. Cribl Edge will attempt to distribute traffic to the connections according to their relative weights.

The final column provides an `X` button to delete any row from the table.

For details on configuring all these options, see About Load Balancing.

When you first enable load balancing, or if you edit the load weight once your data is load-balanced, give the logic time to settle. The changes might take a few seconds to register.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older

than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5` GB. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.

Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Retries

Honor Retry-After header: Toggle on to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream/Edge limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. Any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. Toggle off to ignore all `Retry-After` headers.

Default Retry Behavior: By default, Cribl Stream/Edge applies predefined retry strategies for certain failed requests to the Destination using an exponential backoff algorithm. You can override these settings by manually configuring the retry behavior for a particular error code.

Destinations created before the 4.16.0 release do not support the default `4xx` series retry behavior. If your Destination was created prior to this release and you'd like it to support the behavior, reconfigure the Destination.

To retry requests, Cribl Stream/Edge applies the following rules by default:

Status Code	Action
Any in the `1xx`, `3xx` series	Drop the request

Status Code	Action
`401`, `403`, `408`, `429` errors	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached
Any other `4xx` series	Drop the request
Any in the `5xx` series	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached

By default, any `429 (Too Many Requests)` response that is not listed in the retry settings is treated as a non-retryable `4xx` error and dropped. If you want `429` responses retried, ensure `429` is included in the **Settings for failed HTTP requests**.

Settings for failed HTTP requests: You can also manually configure Cribl Stream/Edge to automatically retry requests that receive particular HTTP response status codes.

Upon receiving a response code that's on the list, Cribl Stream/Edge first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream/Edge continues retrying the request without increasing the time between retries any further.

If the sender (which manages the connection to the Destination) is at capacity, it will not accept any incoming events. These incoming events originate internally from a previous stage of the data flow when Destinations send outbound requests to their respective external services, and they include retry requests and new requests. Any events that were already in transit when the sender reached capacity will continue to be processed downstream.

Sender capacity is freed up when an outgoing request succeeds or encounters a non-retryable error. When the sender has available capacity again, it will resume accepting incoming events. This capacity management is influenced by the number of active connections and configured limits, such as concurrency and buffer sizes. If a Pipeline sends events faster than the Destination can process, the buffers may fill up, leading to backpressure and `Sender at capacity` warnings. This backpressure prevents the sender from accepting additional requests until capacity is restored.

For each response code you want to add to the list, select **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.

- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: Toggle on to automatically retry requests that have timed out and display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Reject certificates that are **not** authorized by a trusted CA (for example, the system's CA). Default is toggled on.

Round-robin DNS: Toggle on to enable round-robin DNS lookup across multiple IP addresses, IPv4 and IPv6. When a DNS server resolves a Fully Qualified Domain Name (FQDN) to multiple IP addresses, Cribl Edge will sequentially use each address in the order they are returned by the DNS server for subsequent connection attempts. (This option is visible only when the **General Settings** > **Load balancing** option is toggled off.)

Compress: Compresses the payload body before sending. Default is toggled on (recommended).

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests per Worker Process. When Cribl Edge hits this limit, it begins throttling traffic to the downstream service. Defaults to `5`. Minimum: `1`. Maximum: `32`.

Body size limit (KB): Maximum size of the request body before compression. Defaults to `4096` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Body size limit** value until downstream receivers reliably accept all events.

Events-per-request limit: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (s): Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Extra HTTP headers: Name-value pairs to pass as additional HTTP headers. Values will be sent encrypted.

Extra parameters: Name-value pairs to pass as additional parameters. If you are using Elastic ingest pipelines, specify an extra parameter whose name is `pipeline` and whose value is the name of your pipeline, similar to these examples.

Elastic version: Determines how to format events. For Elastic Cloud, you must explicitly set version `7.x`. For other Elasticsearch clusters, the `Auto` default will discover the downstream Elasticsearch version automatically, but you have the option to explicitly set version `6.x` or `7.x`.

Elastic pipeline: To send data to an Elastic Ingest pipeline, optionally enter that pipeline's name as a constant. Or, enter a JavaScript expression that evaluates outgoing events and sends matching events to the desired Elastic Ingest pipeline(s). Cribl Edge will first interpret your entry as a constant and try to find a matching value in the event. If it finds no matching value, it will evaluate your **Elastic pipeline** entry as an expression.

For example, the expression `sourcetype=='access_common'?'cribl_pipeline':undefined` matches events whose `sourcetype` is `access_common`, and sends them to an Elastic Ingest pipeline named `cribl_pipeline`.

You can also specify the name of the pipeline in an event field. For example, `myPipelineField` would use the value from the event's `myPipelineField` property (if present) to identify the Elastic Ingest pipeline to process the event. Alternately, the expression `myPipelineField != null ? myPipelineField : undefined` would also identify this field. If the event does not contain such a field, `myPipelineField != null ? myPipelineField : 'theDefaultIndex'` will provide a default index. With this approach, you can override the Elastic Ingest pipeline at the event level.

See also the Elasticsearch Source documentation.



The next two fields appear only when you toggle **Load balancing** on in the **General Settings**.

DNS resolution period (seconds): Re-resolve any hostnames each time this interval recurs, and pick up destinations from the A records. Defaults to `600` seconds.

Load balance stats period (seconds): Lookback traffic history period. Defaults to `300` seconds.

Include document_id: Toggle off to omit the `document_id` field when sending events to an Elastic TSDS (time series data stream).

Write action: Action to use when writing events. Set this option to `Create` (default) when writing to a data stream, which is append-only. Set this option to `Index` only when you write directly to an index and need to update existing records. `Index` will fail if you use it to write to a data stream.

Failed request logging mode: Determines which data is logged when a request fails. Use the drop-down to select one of these options:

- `None` (default).

- `Payload`.
- `Payload + Headers`. Use the **Safe Headers** field below to specify the headers to log. If you leave that field empty, all headers are redacted, even with this setting.

Safe headers: List the headers you want to log, in plain text.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Field Normalization

When sending data to Elasticsearch, this Destination applies the following field normalization to align with the Elasticsearch schema:

- Renames `_time` to `@timestamp`, maintaining millisecond precision.
- Renames `host` to `host.name`.

See also our Elasticsearch Source documentation's **Field Normalization** section.

Internal Fields

Cribl Edge uses a set of internal fields to assist in forwarding data to a Destination.

Fields for this Destination:

- `__id`
- `__type`
- `__index`
- `__host`

Notes on HTTP-Based Outputs

- To proxy outbound HTTP/S requests, see System Proxy Configuration.
- Cribl Edge will attempt to use keepalives to reuse a connection for multiple requests. After two minutes of the first use, the connection will be thrown away, and a new one will be reattempted. This is to prevent sticking to a particular destination when there is a constant flow of events.
- If the server does not support keepalives (or if the server closes a pooled connection while idle), a new connection will be established for the next request.

- When resolving the Destination's hostname with load balancing disabled, Cribl Edge will pick the first IP in the list for use in the next connection. Enable Round-robin DNS to better balance distribution of events between Elasticsearch cluster nodes.

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

12.3.5.2. Elastic Cloud Destination

Cribl Edge can send events to Elastic Cloud.

Use the Elasticsearch Destination if you need flexibility to support both self-hosted and Elastic Cloud deployments and require advanced configuration options.

Type: Streaming | TLS Support: Yes | PQ Support: Yes

For an example of using Cribl Edge to parse and enrich inbound events for Elasticsearch, see our Splunk to Elasticsearch topic.

Configure Cribl Edge to Output to Elastic Cloud

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this Elastic Cloud Destination definition.
    - **Description**: Optionally, enter a description.
    - **Cloud ID**: Enter the Cloud ID of the Elastic Cloud environment where you want to send events. You'll find it on the Deployments overview page of your Elastic Cloud environment.
    - **Data stream or index**: Enter a JavaScript expression that evaluates to the name of the Elastic data stream or Elastic index where you want events to go. The expression is evaluated for each event, can evaluate to a constant value, and must be enclosed in quotes or backticks. An event's `__index` field can overwrite the index or data stream name.
    - **Elastic pipeline**: Enter the name of the Elastic ingest pipeline (optional). You can use an expression to override the pipeline attribute received from the Elasticsearch Source.
3. Under **Authentication**, select an **Authentication method** from the dropdown:
    - **Manual**: Enter your credentials directly in the resulting **Username** and **Password** fields.

- **Secret**: Exposes a **Credentials secret** drop-down, in which you can select a stored secret that references the credentials described above. A **Create** link is available to store a new, reusable secret.
- **Manual API Key**: Exposes an **API key** field to directly enter your Elasticsearch API key.
- **Secret API Key**: Exposes an **API key (text secret)** drop-down, in which you can select a stored text secret that references your Elasticsearch API key. A **Create** link is available to store a new, reusable secret.

4. Next, you can configure the following **Optional Settings**:
   - **Backpressure behavior**: Specify whether to block, drop, or queue events when all receivers are exerting backpressure. Defaults to `Block`.
   - **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
5. Optionally, you can adjust the Persistent Queue, Processing, Retries and Advanced settings outlined in the sections below.
6. Select **Save**, then **Commit & Deploy**.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.



If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5 GB`. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.



Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Retries

Honor Retry-After header: Toggle on to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream/Edge limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. Any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. Toggle off to ignore all `Retry-After` headers.

Default Retry Behavior: By default, Cribl Stream/Edge applies predefined retry strategies for certain failed requests to the Destination using an exponential backoff algorithm. You can override these settings by

manually configuring the retry behavior for a particular error code.



Destinations created before the 4.16.0 release do not support the default `4xx` series retry behavior. If your Destination was created prior to this release and you'd like it to support the behavior, reconfigure the Destination.

To retry requests, Cribl Stream/Edge applies the following rules by default:

Status Code	Action
Any in the `1xx, 3xx` series	Drop the request
`401, 403, 408, 429` errors	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached
Any other `4xx` series	Drop the request
Any in the `5xx` series	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached

By default, any `429 (Too Many Requests)` response that is not listed in the retry settings is treated as a non-retryable `4xx` error and dropped. If you want `429` responses retried, ensure `429` is included in the **Settings for failed HTTP requests**.

Settings for failed HTTP requests: You can also manually configure Cribl Stream/Edge to automatically retry requests that receive particular HTTP response status codes.

Upon receiving a response code that's on the list, Cribl Stream/Edge first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream/Edge continues retrying the request without increasing the time between retries any further.

If the sender (which manages the connection to the Destination) is at capacity, it will not accept any incoming events. These incoming events originate internally from a previous stage of the data flow when Destinations send outbound requests to their respective external services, and they include retry requests and new requests. Any events that were already in transit when the sender reached capacity will continue to be processed downstream.

Sender capacity is freed up when an outgoing request succeeds or encounters a non-retryable error. When the sender has available capacity again, it will resume accepting incoming events. This capacity management is influenced by the number of active connections and configured limits, such as concurrency and buffer sizes. If a Pipeline sends events faster than the Destination can process, the buffers may fill up, leading to backpressure

and `Sender at capacity` warnings. This backpressure prevents the sender from accepting additional requests until capacity is restored.

For each response code you want to add to the list, select **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: Toggle on to automatically retry requests that have timed out and display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests per Worker Process. When Cribl Edge hits this limit, it begins throttling traffic to the downstream service. Defaults to `5`. Minimum: `1`. Maximum: `32`.

Body size limit (KB): Maximum size of the request body before compression. Defaults to `4096` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Body size limit** value until downstream receivers reliably accept all events.

Events-per-request limit: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (s): Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Extra HTTP headers: Name-value pairs to pass as additional HTTP headers. Values will be sent encrypted.

Extra parameters: Name-value pairs to pass as additional parameters. If you are using Elastic ingest pipelines, specify an extra parameter whose name is `pipeline` and whose value is the name of your pipeline, similar to these examples.

Include document_id: Toggle off to omit the `document_id` field when sending events to an Elastic TSDS (time series data stream).

Failed request logging mode: Determines which data is logged when a request fails. Use the drop-down to select one of these options:

- `None` (default).
- `Payload`.
- `Payload + Headers`. Use the **Safe Headers** field below to specify the headers to log. If you leave that field empty, all headers are redacted, even with this setting.

Safe headers: List the headers you want to log, in plain text.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Field Normalization

This Destination normalizes the following fields:

- `_time` becomes `@timestamp` at millisecond resolultion.
- `host.name` is set to `host`.

See also our Elasticsearch Source documentation's **Field Normalization** section.

Internal Fields

Cribl Edge uses a set of internal fields to assist in forwarding data to a Destination.

Fields for this Destination:

- `__id`
- `__type`
- `__index`

- `__host`

Notes on HTTP-Based Outputs

- To proxy outbound HTTP/S requests, see System Proxy Configuration.
- Cribl Edge will attempt to use keepalives to reuse a connection for multiple requests. After two minutes of the first use, the connection will be thrown away, and a new one will be reattempted. This is to prevent sticking to a particular destination when there is a constant flow of events.
- If the server does not support keepalives (or if the server closes a pooled connection while idle), a new connection will be established for the next request.

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

12.3.6. Google Cloud

You can send data from Cribl Edge to the following Google Cloud services:

Destination	Description
Google SecOps	Send data to Google Security Operations (SecOps)
Google Cloud Chronicle API	Send to Google Cloud Chronicle API
Google Cloud Logging	Send data to Google Cloud Logging
Google Cloud Pub/Sub	Send data to Google Cloud Pub/Sub
Google Cloud Storage	Send data to Google Cloud Storage

12.3.6.1. Google Security Operations (SecOps) Destination

Cribl Edge supports sending data to Security Operations (SecOps), a cloud service for retaining, analyzing, and searching enterprise security and network telemetry data. This service was formerly known as Google Chronicle.

If you plan to use **V2** of the Google Security Operations API, you need your Google service account credentials to define a Google SecOps Destination.

If you plan to use **V1** of the Google Security Operations API, Cribl recommends using Google service account credentials for authentication, but alternatively, you can opt to use an API key from Google.

To add labels to individual events instead of at the batch level, you can use the Google Cloud Chronicle API Destination.



Type: Streaming | TLS Support: Yes | PQ Support: Yes

Options for Sending Data to SecOps

When sending data to SecOps, you have two primary methods, each with its own advantages:

- **Sending Unstructured Data for SecOps Parsing**: This is ideal for well-known data types. You send your raw, unstructured data, and SecOps uses its built-in parsers to transform it into the Unified Data Model (UDM). However, this approach can lead to parsing errors and troubleshooting if you're dealing with less common or proprietary data types.
- **Sending Pre-Structured UDM Data**: This method bypasses SecOps's parsing process entirely. It's especially beneficial for less-usual or unique, proprietary data types that might confuse SecOps's parsers. By sending data that's already structured as UDM, you gain greater control and flexibility over how your data lands in SecOps. The trade-off is that you "own" the data transformation process, using Pipelines and Functions to convert your initial unstructured data to UDM before sending it to SecOps.
- **Sending Pre-Structured UDM Entity Data**: This is an extension of the **Pre-Structured UDM Data** option, specifically for sending entity data to the `v2/entities:batchCreate` endpoint. This data is used to enrich logs and provide additional context on events. Like sending UDM logs, you must format your events to conform to the UDM entity schema before sending them to SecOps.



Do not place UDM fields like `metadata`, `principal`, and `network` inside `_raw`. These fields must exist as top-level fields in the event payload.

Working with Unstructured Log Types

Google SecOps supports a dynamic list of unstructured log types that evolves over time. There are two configuration settings for managing log type assignments:

- **Default log type**: Cribl Edge periodically updates its **Default log type** dropdown with the latest list from Google.
- **Custom log types**: If a log type you need isn't in the default dropdown, or if you're using an older, removed log type, you can define and manage it in the **Custom log types** table under **Optional Settings**.

For Google's latest list of supported log types, see the Google SecOps SIEM reference or send a `GET` request to `https://malachiteingestion-pa.googleapis.com/v1/logtypes`.

How Log Types are Assigned

- **Event-specific**: When processing an event, Cribl Edge first checks if the event has a `__logType` field. If this field is present, its value determines the log type for that event.
- **Default**: If the event lacks a `__logType` field, Cribl Edge assigns the log type specified in the **Default log type** drop-down.
- **Custom**: If the **Default log type** is set to `Custom`, Cribl Edge will use the first custom log type listed in the **Custom log types** table. Reorder the custom log types in the table using the grab handles to prioritize which custom log type should be used as the default.
    - If the **Default log type** is set to a specific log type (not `Custom`), the custom log types defined in the **Custom log types** table will not be used as defaults. However, you can still use these custom log types by explicitly setting the `__logType` field in your events to one of the custom log types.

Configure Cribl Edge to Output to SecOps

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:

- **Output ID**: Enter a unique name to identify this SecOps output definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
- **Description**: Optionally, enter a description.
- **API version**: The version of the Google Security Operations Ingestion API you want the Destination to use when communicating with SecOps. Defaults to `V2`.
- **Send events as**: Choose how you want the Destination to send data to SecOps. The options are: **Unstructured** and **UDM**. See Options for Sending Data to SecOps to learn about the differences between working with `Unstructured` and already-structured `UDM` data.

3. Next, configure the settings that appear based on your **API version** and **Send events** as selections. For detail, see the API Version and Output Format Options section.
4. Next, you can configure the following **Optional Settings**: (The following settings are available regardless of how **General Settings** > **Send events as** is set.)
   - **Region**: The Google SecOps regional endpoint to send events to.
   - **Backpressure behavior**: Whether to block, drop, or queue events when all receivers are exerting backpressure. (Causes might include a broken or denied connection, or a rate limiter.) Defaults to `Block`.
   - **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
5. Optionally, you can adjust the Authentication, Persistent Queue, Processing, Retries and Advanced settings outlined in the sections below.
6. Select **Save**, then **Commit & Deploy**.

API Version and Output Format Options

The settings that appear in the **New Destination** modal depend on your selections for **API version** and **Send events as**. The following sections describe how these choices affect the available configuration options.



For Google SecOps unstructured parsers, set the Destination **Log text field** to `_raw`. This ensures that the payload body is the raw log line and that fields like `message_trace_id` are at the payload root.

API version: V2 (Default)

The API version `V2` is the recommended version for most use cases.

- If you choose **Send events as**: `UDM`, an additional **UDM type** option displays. You must select either `Logs` for standard log data or `Entities` for entity-specific data.
- If you choose **Send events as**: `Unstructured`, the following settings appear under **General Settings**:

- **Default log type**: Select a log type to assign to events that lack a `__logType` field.
- **Customer ID**: A unique identifier (UUID) corresponding to a particular SecOps instance.

The following are **Optional Settings**:

- **Custom log types**: For each log type you want to define, select **Add type** and enter values for **Log type** and **Description** in the resulting table row. See Working with Unstructured Log Types above for more details.
- **Namespace**: Configure an environment namespace to identify logs' originating data domain. You can use this as a tag when indexing and/or enriching data. The `__namespace` event field, if present, will overwrite this.
- **Custom labels**: Define labels for the Destination to add to events. These are used by Google SecOps for features such as data RBAC and filtering. See how to configure data RBAC for reference lists. Select **Add label** to create a new label and specify it as a key-value pair.
- **Log text field**: Set to the event field containing the raw log string (commonly `_raw`). If left blank, the Destination sends a JSON representation of the whole event. Use this when you want Google to parse the raw line itself rather than JSON you provide.

API version V1

THe API version `V1` is a legacy version. We recommend using `V2` for all new deployments, but `V1` is available for backward compatibility with existing configurations or for environments that have specific requirements that prevent them from using `V2`.

- If you choose **Send events as**: `UDM`, no additional settings appear, but the **Custom labels** setting is available.
- If you choose **Send events as**: `Unstructured`, the following settings appear under **General Settings**:
  - **Default log type**: Select a log type for events that lack a `__logType` field.

  The following are **Optional Settings**:
  - **Custom log types**: For each log type you want to define, select **Add type** and enter values for **Log type** and **Description** in the resulting table row. See Working with Unstructured Log Types above for more details.
  - **Namespace**: Configure an environment namespace to identify logs' originating data domain. You can use this as a tag when indexing and/or enriching data. The `__namespace` event field, if present, will overwrite this.
  - **Customer ID**: A unique identifier (UUID) corresponding to a particular SecOps instance.
  - **Custom labels**: Define labels for the Destination to add to events. These are used by Google SecOps for features such as data RBAC and filtering. See how to configure data RBAC for reference lists. Select **Add label** to create a new label and specify it as a key-value pair.

- **Log text field**: Set to the event field containing the raw log string (commonly `_raw`). If left blank, the Destination sends a JSON representation of the whole event. Use this when you want Google to parse the raw line itself rather than JSON you provide.

Authentication

To complete this part of the Destination definition, you will need either a Google Security Operations API key or your Google SecOps service account credentials. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.

Use the **Authentication method** to select an option. Whether you have set **General Settings > API version** to `V2` or `V1` determines what options are displayed.

For API version `V2`, the options are:

- **Service Account Credentials**: This option exposes a field where you can enter your Google SecOps service account directly.
- **Service Account Credentials Secret**: This option exposes a drop-down, in which you can select a stored secret that references your Google SecOps service account. A **Create** link is available to store a new, reusable secret.

For API version `V1`, the options include the two listed above plus two more:

- **API key**: This option exposes a field where you can enter your Google Security Operations API key.
- **API Key Secret**: This option exposes a drop-down, in which you can select a stored secret that references your Google Security Operations API key. A **Create** link is available to store a new, reusable secret.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk

> space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.
>
> This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

> If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5` GB. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.

- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.



Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Processing Settings Post-Processing

Pipeline: In this section's **Pipeline** drop-down list, you can select a single existing Pipeline or Pack to process data from this input before the data is sent through the Routes.

System fields: Specify any fields you want Cribl Edge to automatically add to events using this output. Wildcards are supported. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Retries

Honor Retry-After header: Whether to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Edge limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. When enabled, any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. When disabled, all `Retry-After` headers are ignored.

Settings for failed HTTP requests: When you want to automatically retry requests that receive particular HTTP response status codes, use these settings to list those response codes.

For any HTTP response status codes that are not explicitly configured for retries, Cribl Edge applies the following rules:

Status Code	Action
Any in the `1xx`, `3xx`, or `4xx` series	Drop the request
Any in the `5xx` series	Retry the request

Upon receiving a response code that's on the list, Cribl Edge first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Edge continues retrying the request without increasing the time between retries any further.

If the sender (which manages the connection to the Destination) is at capacity, it will not accept any incoming events. These incoming events originate internally from a previous stage of the data flow when Destinations send outbound requests to their respective external services, and they include retry requests and new requests. Any events that were already in transit when the sender reached capacity will continue to be processed downstream.

Sender capacity is freed up when an outgoing request succeeds or encounters a non-retryable error. When the sender has available capacity again, it will resume accepting incoming events. This capacity management is influenced by the number of active connections and configured limits, such as concurrency and buffer sizes. If a Pipeline sends events faster than the Destination can process, the buffers may fill up, leading to backpressure and `Sender at capacity` warnings. This backpressure prevents the sender from accepting additional requests until capacity is restored.

By default, `429 (Too Many Requests)` and `503 (Service Unavailable)` are the only response codes configured for automatic retries. For each response code you want to add to the list, select **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `30000` (30 seconds).
- **Backoff multiplier**: The base for the exponential backoff algorithm; defaults to `1`. A value of `2` means that Cribl Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Edge should apply for its final retry, in milliseconds. Defaults to `30000` (30 seconds); minimum is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: When you want to automatically retry requests that have timed out, toggle this control on to display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Reject certificates that are **not** authorized by a CA in the **CA certificate path**, or by another trusted CA (such as the system's CA). Defaults to toggled on.

Round-robin DNS: Toggle on to enable round-robin DNS lookup across multiple IP addresses, IPv4 and IPv6. When a DNS server resolves a Fully Qualified Domain Name (FQDN) to multiple IP addresses, Cribl Edge will sequentially use each address in the order they are returned by the DNS server for subsequent connection attempts.

Compress: Toggle on (default) to compress the payload body before sending.

Request timeout: Enter an amount of time, in seconds, to wait for a request to complete before aborting it. Defaults to `90`.

Request concurrency: Enter the maximum number of ongoing requests to allow before blocking.

Body size limit (KB): Maximum size of the request body before compression. Defaults to `1024` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Body size limit** value until downstream receivers reliably accept all events.

Buffer memory limit (KB): Total amount of memory used to buffer outgoing requests waiting to be sent. If left blank, defaults to five times the max body size (if set). If `0`, no limit is enforced. This provides granular control

over the memory allocated for buffering outgoing batched requests. Increasing the limit allows batches to grow larger before being flushed, improving efficiency for data with high cardinality (a large number of unique batches). Finding the optimal balance between efficient data transfer and memory usage involves adjusting both the **Buffer memory limit** and **Body size limit** settings.

Events-per-request limit: Enter the maximum number of events to include in the request body. Defaults to `0` (unlimited).

Flush period (sec): Enter the maximum time to allow between requests. Be aware that small values could cause the payload size to be smaller than the configured **Body size limit**.

Extra HTTP Headers: Select **Add Header** to insert extra headers as **Name/Value** pairs. Values will be sent encrypted.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Cribl Edge will redact all headers, except non-sensitive headers that you declare below in **Safe headers**.

Safe headers: Add headers to declare them as safe to log in plaintext. (Sensitive headers such as `authorization` will always be redacted, even if listed here.) Use a tab or hard return to separate header names.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Proxying Requests

If you need to proxy HTTP/S requests, see System Proxy Configuration.

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

12.3.6.2. Google Cloud Chronicle API Destination

The **Google Cloud Chronicle API** Destination sends data to Google Cloud Chronicle using the v1alpha `ImportLogs` ingestion method. The Destination sends your raw, unstructured data, and Chronicle API uses its built-in parsers to transform it into the Unified Data Model (UDM).

Using the **Google Cloud Chronicle API** Destination, in contrast to the Google Security Operations (SecOps) Destination, lets you add labels to individual events, not only at the batch level, and take advantage of the improved performance and scaling offered by Chronicle API.



Type: Streaming | TLS Support: Yes | PQ Support: Yes

Working with Unstructured Log Types

Google Cloud Chronicle API supports a dynamic list of unstructured log types that evolves over time.

The **Default log type** setting lets you select the log type to send. To send a log of an unsupported type, type your custom log type into this field.

For Google's latest list of supported log types, see the Supported log types and default parsers in Google Cloud documentation to learn more about the specific log types.

How Log Types are Assigned

- **Event-specific**: When processing an event, Cribl Edge first checks if the event has a `__logType` field. If this field is present, its value determines the log type for that event.
- **Default**: If the event lacks a `__logType` field, Cribl Edge assigns the log type specified in the **Default log type** drop-down.
- **Custom**: If you entered a custom log name into **Default log type**, Cribl Edge will use this type.

Configure a Google Cloud Chronicle API Destination

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select**

Existing or **Add New**.

- To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.

2. In the **New Destination** modal, configure the following under **General Settings**:
   - **Output ID**: Enter a unique name to identify this Destination. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
   - **Description**: Optionally, enter a description.
   - **Default log type**: Select or type in the default log type value to assign to events that lack a `__logType` field. See Working with Unstructured Log Types for more information.
   - **GCP project ID**: Enter the Google Cloud Platform (GCP) project ID to send events to. You can find the project ID on the Google Cloud Welcome page.
   - **GCP instance**: Enter ID of the GCP instance to send events to. This is the Chronicle customer UUID.
   - **Region**: Select the regional endpoint to send events to.
   - **Log text field**: Optionally, select the name of the event field that contains the log text to send. If not specified, Cribl Edge sends a JSON representation of the whole event.
3. Next, you can configure the following **Optional Settings**:
   - **Namespace**: User-configured environment namespace to identify the data domain from which the logs originated. This namespace is used as a tag to identify the appropriate data domain for indexing and enrichment functionality. The `__namespace` event field, if present, will overwrite this.
   - **Custom labels**: Custom labels to be added to every event. Custom labels are a Google SecOps feature used for data role-based access control (data RBAC) and other purposes. See the Google SecOps documentation. Click **Add label** to create a new label and specify it as a key-value pair.
     - **Enable RBAC**: Toggle on to designate a the custom label for RBAC and filtering, allowing granular multi-tenant data management directly within Chronicle. The label is flagged in the event payload, and Chronicle enforces access boundaries based on its value. This enables per-event RBAC, so you can dynamically tag events and restrict visibility to authorized roles without duplicating Destinations or relying on parser extensions. The label must resolve for all events in a batch.
   - **Backpressure behavior**: How to handle events when all receivers are exerting backpressure.
   - **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
4. Under **Authentication**, select an **Authentication method** from the dropdown:
   - **Service Account Credentials**: This option exposes a field where you can enter your Google SecOps service account directly.

- **Service Account Credentials Secret**: This option exposes a drop-down, in which you can select a stored secret that references your Google SecOps service account. A **Create** link is available to store a new, reusable secret.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older

than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5` GB. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.

Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Retries

Honor Retry-After header: Toggle on to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream/Edge limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. Any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. Toggle off to ignore all `Retry-After` headers.

Default Retry Behavior: By default, Cribl Stream/Edge applies predefined retry strategies for certain failed requests to the Destination using an exponential backoff algorithm. You can override these settings by manually configuring the retry behavior for a particular error code.

Destinations created before the 4.16.0 release do not support the default `4xx` series retry behavior. If your Destination was created prior to this release and you'd like it to support the behavior, reconfigure the Destination.

To retry requests, Cribl Stream/Edge applies the following rules by default:

Status Code	Action
Any in the `1xx`, `3xx` series	Drop the request

Status Code	Action
`401`, `403`, `408`, `429` errors	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached
Any other `4xx` series	Drop the request
Any in the `5xx` series	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached

By default, any `429 (Too Many Requests)` response that is not listed in the retry settings is treated as a non-retryable `4xx` error and dropped. If you want `429` responses retried, ensure `429` is included in the **Settings for failed HTTP requests**.

Settings for failed HTTP requests: You can also manually configure Cribl Stream/Edge to automatically retry requests that receive particular HTTP response status codes.

Upon receiving a response code that's on the list, Cribl Stream/Edge first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream/Edge continues retrying the request without increasing the time between retries any further.

If the sender (which manages the connection to the Destination) is at capacity, it will not accept any incoming events. These incoming events originate internally from a previous stage of the data flow when Destinations send outbound requests to their respective external services, and they include retry requests and new requests. Any events that were already in transit when the sender reached capacity will continue to be processed downstream.

Sender capacity is freed up when an outgoing request succeeds or encounters a non-retryable error. When the sender has available capacity again, it will resume accepting incoming events. This capacity management is influenced by the number of active connections and configured limits, such as concurrency and buffer sizes. If a Pipeline sends events faster than the Destination can process, the buffers may fill up, leading to backpressure and `Sender at capacity` warnings. This backpressure prevents the sender from accepting additional requests until capacity is restored.

For each response code you want to add to the list, select **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.

- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: Toggle on to automatically retry requests that have timed out and display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Reject certificates that are not authorized by a trusted CA (for example, the system's CA). Defaults to toggled on.

Round-robin DNS: Toggle on to enable round-robin DNS lookup across multiple IP addresses, IPv4 and IPv6. When a DNS server resolves a Fully Qualified Domain Name (FQDN) to multiple IP addresses, Cribl Edge will sequentially use each address in the order they are returned by the DNS server for subsequent connection attempts.

Compress: Toggle on (default) to compress the payload body before sending.

Endpoint: The Chronicle API service endpoint. If empty, this defaults to the endpoint in the **Region** field. Otherwise, it must point to a Chronicle API-compatible endpoint (for example, https://custom-endpoint.googleapis.com).

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests per Worker Process. When Cribl Edge hits this limit, it begins throttling traffic to the downstream service. Defaults to `5`. Minimum: `1`. Maximum: `32`. Each request can potentially hit a different HEC receiver.

Body size limit (KB): Maximum size, in KB, of the request body. Defaults to `5120`. Lowering the size can potentially result in more parallel requests and also cause outbound requests to be made sooner.

Events-per-request limit: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values can cause the payload size to be smaller than the configured **Body size limit**. Defaults to `5`.

- Retries happen on this flush interval.
- Any HTTP response code in the `2xx` range is considered success.
- Any response code in the `5xx` range is considered a retryable error, which will not trigger persistent queue (PQ) usage.
- Any other response code will trigger PQ (if PQ is configured as the Backpressure behavior).

Extra HTTP headers: Select **Add Header** to add **Name**/**Value** pairs to pass as additional HTTP headers. Values will be sent encrypted.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Cribl Edge will redact all headers, except non-sensitive headers that you declare below in **Safe headers**.

Safe headers: Add headers to declare them as safe to log in plaintext. (Sensitive headers such as `authorization` will always be redacted, even if listed here.) Use a tab or hard return to separate header names.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Proxying Requests

If you need to proxy HTTP/S requests, see System Proxy Configuration.

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

12.3.6.3. Google Cloud Logging Destination

Cribl Edge supports sending log data to Google Cloud Logging, a real-time log storage and management service with search, analysis and alerting.



Type: Streaming | TLS Support: Yes | PQ Support: Yes

Configure Cribl Edge to Output to Google Cloud Logging

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this Google Cloud Logging definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
    - **Description**: Optionally, enter a description.
    - **Log location type**: Select an option from the dropdown to determine where your log entries are categorized. Choose **Project**, **organization**, **Billing Account**, or **Folder**. For details, see Log Location Types.
    - **Log name expression**: Enter a JavaScript expression to compute the log name's value.
    - **Validate and correct log name**: Toggle on to ensure that log names are sanitized and truncated for compliance and reliability. Log name expression characters that do not meet Google requirements are replaced by an underscore `_`. Additionally, the log name is truncated to a maximum of 512 characters, which is the upper limit enforced by Google Cloud Logging. Without sanitization, Google Cloud Logging will reject requests with invalid log names, resulting in dropped events and potential data loss.
3. Under **Field Mappings**, configure the following:
    - **Payload format**: Select an option from the dropdown specify how data is structured within a log entry. Choose `Text` or `JSON`. For details, see Field Mappings



This Destination compresses all payloads, using gzip.

- **Log labels**: Click **Add label** to apply one or more labels to log entries. On each row of the resulting table, define each label with:
- **Label**: Arbitrary name for the label.
- **Value**: JavaScript expression to compute the label's value.
- **Resource type expression**: JavaScript expression to compute the field value for the managed resource type. Must evaluate to one of the monitored resource types listed here. If not specified, falls back to `global`.
- **Resource labels**: Click **Add label** to apply one or more labels to the managed resource. For details, see Resource Labels.
- **Severity expression**: JavaScript expression to compute the severity field's value. Must evaluate to one of the `LogSeverity` values listed here. Case-insensitive. If not specified, falls back to `DEFAULT`.
- **Insert ID expression**: JavaScript expression to compute the insert ID field's value.

4. Next, you can configure the following **Optional Settings**:
   - **Backpressure behavior**: Whether to block, drop, or queue events when all receivers are exerting backpressure. Defaults to `Block`.
   - **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
5. To add **Indexed Fields** to your LogEntry objects, provide JavaScript expressions to populate fields beyond the standard LogEntry structure, enabling richer analysis and correlation within Chronicle. For each field you want to index, provide a JavaScript expression. Ensure the expression's result matches the expected data type defined in the LogEntry documentation. Mismatched types prevent the field from appearing in Chronicle.
   - You can configure the following indexed fields using JavaScript expressions: **Trace expression**, **Span ID expression**, **Trace sampled expression**, **HTTP Request**, **Operation**, **Source Location**, and **Log Split**.



The LogEntry object's `timestamp` field is populated from the event's `_time` field.

6. Optionally, you can adjust the Authentication, Processing, and Advanced settings outlined in the sections below.
7. Select **Save**, then **Commit & Deploy**.

Log Location Types

Log Location Types allow you to categorize your log entries based on organizational structures. Choose one of the following options to associate log entries with a specific project, organization, billing account, or folder:

- `Project`: Displays a **Project ID expression** field. Here, enter a JavaScript expression to compute the value of the project ID to associate log entries with.
- `Organization`: Displays a **Organization ID expression** field. Enter a JS expression to compute the value of the organization ID to associate log entries with.
- `Billing Account`: Displays a **Billing account ID expression** field. Enter a JS expression to compute the value of the billing account ID to associate log entries with.
- `Folder`: Displays a **Folder expression** field. Enter a JS expression to compute the value of the folder ID to associate log entries with.

Field Mappings

Field Mappings specify how log entries should be structured. Choose one of the following options:

- `Text`: This default option displays a **Payload text expression** field. Here, enter JavaScript expression to compute the payload's value. Must evaluate to a JavaScript string value; if validation fails, this will default to a JSON string representation of the event.
- `JSON`: Displays a **Payload object expression** field. Enter a JS expression to compute the payload's value. Must evaluate to a JavaScript object value; if validation fails, this will default to the entire event.

Resource Labels

Each **Resource label** must correspond to a valid label for the resource type you've specified in the **Resource type expression**, or else Google Cloud Logging will drop it. On each row of the resulting table, define each label with:

- **Label**: Arbitrary name for the label.
- **Value**: JavaScript expression to compute the label's value.

Authentication

Use the **Authentication method** drop-down to select one of these options:

- **Auto**: This option uses the `GOOGLE_APPLICATION_CREDENTIALS` environment variable, and requires no configuration here.
- **Manual**: This default option displays a **Service account credentials** field for you to enter the contents of your service account credentials file (a set of JSON keys), as downloaded from Google Cloud. To insert the file itself, click the upload button at this field's upper right. As an alternative, you can use environment variables, as outlined here.

- **Secret**: This option exposes a drop-down in which you can select a stored secret that references the service account credentials described above. A **Create** link is available to store a new, reusable secret.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.



If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5` GB. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.

Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Advanced Settings

Request concurrency: Maximum number of concurrent requests before blocking. This is set per Worker Process. Defaults to `5`.

Connection timeout: Amount of time (in milliseconds) to wait for the connection to establish before retrying. Defaults to `10000` (10 sec.).

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30` sec.

Body size limit (KB): Maximum size of the request body before compression. Defaults to `4096` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Body size limit** value until downstream receivers reliably accept all events.

Buffer memory limit (KB): Total amount of memory used to buffer outgoing requests waiting to be sent. If left blank, defaults to 5 times the max body size (if set). If 0, no limit is enforced. This provides granular control over the memory allocated for buffering outgoing batched requests. Increasing the limit allows batches to grow larger before being flushed, improving efficiency for data with high cardinality (a large number of unique batches). Finding the optimal balance between efficient data transfer and memory usage involves adjusting both the **Buffer memory limit** and **Body size limit** settings.

Events-per-request limit: Maximum number of events to include in the request body. Defaults to `0` (unlimited).

Flush period (sec): Maximum time between requests. Low values could cause the payload size to be smaller than the configured **Max body size**. Defaults to `1` sec.

Throttle request rate: Enter a maximum number of requests to send per second, as necessary to comply with Google Cloud's API limits. This field defaults to empty (no throttling), and accepts throttling rates as high as `2000` requests per second.

Environment: Optionally, specify a single Git branch on which to enable this configuration. If this field is empty, the config will be enabled everywhere.

Google Cloud Roles and Permissions

Your Google Cloud service account must have at least the following role:

- `Logs Writer`

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

12.3.6.4. Google Cloud Pub/Sub Destination

Cribl Edge supports sending data to Google Cloud Pub/Sub, a managed real-time messaging service for sending and receiving messages between applications.



Type: Streaming | TLS Support: Yes | PQ Support: Yes

Prerequisites

When working with Google Cloud Pub/Sub, you need to know:

- The project where your credentials originated.
- The project that contains any topic you want to work with.

If the desired topic resides in the same project where your credentials originated, describe it using either its short name **or** its fully-qualified name.

If the desired topic resides in a different project than the one where your credentials originated, you **must** describe it using its fully-qualified name.

Here's an example of a fully-qualified topic name: `projects/my-project-id/topics/my-topic-id`.

The short name for the same topic would be: `my-topic-id`.

Configure Cribl Edge to Output to Pub/Sub

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this Pub/Sub output definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.

- **Description**: Optionally, enter a description.
- **Topic ID**: ID of the Pub/Sub topic to send events to. This static initial configuration of the topic is required; but you can optionally also override it dynamically on a per-event basis.

3. Next, you can configure the following **Optional Settings**:

- **Create topic**: Toggle on if you want Cribl Edge to create the topic on Pub/Sub if it does not exist.
- **Ordered delivery**: Toggle on if you want Cribl Edge to send events in the order that they arrived in the queue. For ordered delivery to work correctly, ensure that both the process receiving events and the subscription from which messages are read have ordered delivery enabled.
- **Region**: Region to publish messages to. Select `default` to allow Google to auto-select the nearest region. (If you've enabled **Ordered delivery**, the selected region must be allowed by message storage policy.)
- **Backpressure behavior**: Whether to block, drop, or queue events when all receivers are exerting backpressure. Defaults to `Block`.
- **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

4. Optionally, you can adjust the Authentication, Persistent Queue, Processing, and Advanced settings outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

Authentication

Use the **Authentication method** drop-down to select one of these options:

- **Auto**: This option uses the `GOOGLE_APPLICATION_CREDENTIALS` environment variable to provide the location of a credential JSON file. The environment variable can also be set directly in JSON format. When this option is set, Cribl Edge will first check whether the `GOOGLE_APPLICATION_CREDENTIALS` environment variable exists and properly set with valid credentials. If this environmental variable is missing or incorrect, Cribl Edge checks for credentials in two additional fallback locations, as described in the official Google Cloud documentation: How Application Default Credentials works. In addition to using the `GOOGLE_APPLICATION_CREDENTIALS` environment variable, you can use credentials attached to your Google Cloud instance.
- **Manual**: This default option displays a **Service account credentials** field for you to enter the contents of your service account credentials file (a set of JSON keys), as downloaded from Google Cloud.
  To insert the file itself, select the upload button at this field's upper right. As an alternative, you can use environment variables, as outlined here.
- **Secret**: This option exposes a drop-down in which you can select a stored secret that references the service account credentials described above. A **Create** link is available to store a new, reusable secret.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.



If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5` GB. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.



Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Advanced Settings

Batch size: The maximum number of items the Google API should batch before it sends them to the topic. Defaults to `10` items.

Batch timeout (ms): The maximum interval (in milliseconds) that the Google API should wait to send a batch (if the configured **Batch size** limit has not been reached).. Defaults to `100` ms.

Queue size limit: Maximum number of queued batches before blocking. Defaults to `100`.

Batch size limit (KB): Maximum size for each sent batch. Defaults to `256` KB.

Flush period (sec): Enter the maximum time to allow between requests (in seconds). Low values could cause the payload size to be smaller than the configured **Batch size limit (KB)**. Defaults to `1` second.

Concurrent request limit: The maximum number of in-progress API requests before Cribl Edge applies backpressure. Defaults to `10`.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Google Cloud Roles and Permissions

Your Google Cloud service account should have at least the following roles on topics:

- `roles/pubsub.publisher`
- `roles/pubsub.viewer` or `roles/viewer`

To enable Cribl Edge's **Create topic** option, your service account should have one of the following (or higher) roles:

- `roles/pubsub.editor`
- `roles/editor`

Either `editor` role confers multiple permissions, including those from the lower `viewer`, `subscriber`, and `publisher` roles. For additional details, see the Google Cloud Access Control topic.

Proxying Requests

If you need to proxy HTTP/S requests, see System Proxy Configuration.

Let's Change the Topic

The Pub/Sub Destination supports alternate topics specified at the event level in the `__topicOut` field. So (for example) if a Pub/Sub Destination is configured to send to main topic `topic1`, and Cribl Edge receives an event with `__topicOut: topic2`, then Cribl Edge will override the main topic and send this event to `topic2`.

However, a topic specified in the event's `__topicOut` field must already exist on Pub/Sub. If it does not, Cribl Edge cannot dynamically create the topic, and will drop the event. On the Destination's **Status** tab, the **Dropped** metric tracks the number of events dropped because a specified alternate topic did not exist.

(This example uses the short topic name, and that works as long as the topic resides in the project where your GCP credentials originated. If the topic resides in a different project, you must use the fully-qualified topic name, for example `__topicOut: projects/<my-project-id>/topics/topic2`.)

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

12.3.6.5. Google Cloud Storage Destination

Google Cloud Storage is a non-streaming Destination type.



Type: Non-Streaming | TLS Support: Yes | PQ Support: No

Configure Cloud Storage Permissions

Configure access permissions on the Cloud Storage side based on the authentication method you'll use on this Destination:

- **Auto authentication**: Supports both uniform and fine-grained access control.
- **Manual authentication**: Requires fine-grained access control on the buckets because it relies on ACLs to access resources.

The `storage.objects.create` permission is required.

If you enable **Verify if bucket exists** in the **Advanced Settings**, the `storage.objects.list` permission is also required.

Assign either of the following roles to grant the required permissions:

- `roles/storage.admin`
- `roles/storage.legacyBucketWriter`

To grant more limited access, combine these roles:

- `roles/storage.objectCreator` to provide the `storage.objects.create` permission
- `roles/storage.objectViewer` to provide the `storage.objects.list` permission

For details, see the Cloud Storage Overview of Access Control and Understanding Roles documentation.

Configure Cribl Edge to Output to Cloud Storage Destinations

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:

- To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
- To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.

2. In the **New Destination** modal, configure the following under **General Settings**:
   - **Output ID**: Enter a unique name to identify this Cloud Storage definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
   - **Description**: Optionally, enter a description.
   - **Bucket name**: Name of the destination bucket. This value can be a constant, or a JavaScript expression that can be evaluated only at init time. For example, referencing a Global Variable: `myBucket-${C.vars.myVar}`.
   - **Region**: Region where the bucket is located.
   - **Staging location**: Filesystem location in which to locally buffer files before compressing and moving to final destination. Cribl recommends that this location be stable and high-performance.

     In Cribl Stream, the **Staging location** field is not displayed or available on Cribl.Cloud-managed Worker Groups.

   - **Key prefix**: Root directory to prepend to path before uploading. Enter a constant, or a JS expression enclosed in single quotes, double quotes, or backticks.
   - **Data format**: The output data format defaults to `JSON`. `Raw` and `Parquet` are also available. Selecting `Parquet` (supported only on Linux, not Windows) exposes a **Parquet Settings** left tab, where you **must** configure certain options in order to export data in Parquet format.
3. Next, you can configure the following **Optional Settings**:
   - **Partitioning expression**: JavaScript expression that defines how files are partitioned and organized. Default is date-based. If blank, Cribl Edge will fall back to the event's `__partition` field value (if present); or otherwise to the root directory of the **Output Location** and **Staging Location**.
   - **Compression**: Data compression format used before moving to final destination. Defaults to `gzip` (recommended). This setting is not available when **Data format** is set to `Parquet`.
   - **File name prefix expression**: The output filename prefix. Must be a JavaScript expression (which can evaluate to a constant), enclosed in quotes or backticks. Defaults to `CriblOut`.
   - **File name suffix expression**: The output filename suffix. Must be a JavaScript expression (which can evaluate to a constant), enclosed in quotes or backticks. Defaults to `` `.${C.env["CRIBL_WORKER_ID"]}.${__format}${__compression === "gzip" ? ".gz" : ""}` ``, where `__format` can be `json` or `raw`, and `__compression` can be `none` or `gzip`.



To prevent files from being overwritten, Cribl appends a random sequence of 6 characters to the end of their names. This also applies to prefix and suffix expressions in file names.

For example, if you set the **File name prefix expression** to `CriblExec` and set the **File name suffix expression** to `.csv`, the file name might display as `CriblExec-adPRWM.csv` with `adPRWM` appended.

- **Backpressure behavior**: Select whether to block or drop events when all receivers are exerting backpressure. (Causes might include an accumulation of too many files needing to be closed.) Defaults to `Block`.
- **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

4. Optionally, you can adjust the Authentication, Parquet, Processing, Retries and Advanced settings outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

Authentication

Use the **Authentication method** drop-down to select one of these options:

- **Auto**: This option authenticates with the attached Google Cloud Platform (GCP) Service Account, relying on GCP IAM roles to access the appropriate GCP resources. This option is available only when Cribl Edge is on-prem, and the Edge Nodes are running in Google Compute Engine VMs on GCP. Supports both uniform and fine-grained access control.
- **Manual**: With this default option, authentication is via HMAC (Hash-based Message Authentication Code). To create a key and secret, see Google Cloud's Managing HMAC Keys for Service Accounts documentation. This option requires fine-grained access control on the GCS bucket because it relies on ACLs to access resources. Uniform access control is not supported with this method. This option exposes these two fields:
  - **Access key**: Enter the HMAC access key.
  - **Secret key**: Enter the HMAC secret.

The values for **Access key** and **Secret key** can be a constant, or a JavaScript expression (such as `${C.env.MY_VAR}`) enclosed in quotes or backticks, which allows configuration with environment variables.

- **Secret**: This option exposes a **Secret key pair** drop-down, in which you can select a stored secret that references the secret key pair described above. A **Create** link is available to store a new, reusable secret.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports `c*` wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Parquet Settings

To write out Parquet files, note that:

- On Linux, you can use the Cribl Edge CLI's `parquet` command to view a Parquet file, its metadata, or its schema.
- Cribl Edge Workers support Parquet only when running on Linux, not on Windows.
- See Parquet Schemas for pointers on how to avoid problems such as data mismatches.

Automatic schema: Toggle on to automatically generate a Parquet schema based on the events of each Parquet file that Cribl Edge writes. Toggle off (default) to expose the following additional field:

- **Parquet schema**: Select a schema from the drop-down.



If you need to modify a schema or add a new one, follow the instructions in our Parquet Schemas topic. These steps will propagate the freshest schema back to this drop-down.

Parquet version: Determines which data types are supported, and how they are represented. Defaults to `2.6`; `2.4` and `1.0` are also available.

Data page version: Serialization format for data pages. Defaults to `V2`. If your toolchain includes a Parquet reader that does not support `V2`, use `V1`.

Group row limit: The number of rows that every group will contain. The final group can contain a smaller number of rows. Defaults to `10000`.

Page size: Set the target memory size for page segments. Generally, set lower values to improve reading speed, or set higher values to improve compression. Value must be a positive integer smaller than the **Row group size** value, with appropriate units. Defaults to `1 MB`.

Log invalid rows: Toggle on to output up to 20 unique rows that were skipped due to data format mismatch. Log level must be set to `debug` for output to be visible.

Write statistics: Toggle on (default) if you have Parquet tools configured to view statistics - these profile an entire file in terms of minimum/maximum values within data, numbers of nulls, etc.

Write page indexes: Toggle on (default) if your Parquet reader uses statistics from Page Indexes to enable page skipping. One Page Index contains statistics for one data page.

Write page checksum: Toggle on if you have configured Parquet tools to verify data integrity using the checksums of Parquet pages.

Metadata (optional): The metadata of files the Destination writes will include the properties you add here as key-value pairs. For example, one way to tag events as belonging to the OCSF category for security findings would be to set **Key** to `OCSF Event Class` and **Value** to `2001`.

Advanced Settings

File size limit (MB): Maximum uncompressed output file size. Files of this size will be closed and moved to final output location. Defaults to `32`.

File open time limit (sec): Maximum amount of time to write to a file. Files open for longer than this limit will be closed and moved to final output location. Defaults to `300`. **Idle time limit (sec)**: Maximum amount of time to keep inactive files open. Files open for longer than this limit will be closed and moved to final output location. Defaults to `30`. **Open file limit**: Maximum number of files to keep open concurrently. When exceeded, the oldest open files will be closed and moved to final output location. Defaults to `100`.



Cribl Edge will close files when **either** of the `File size limit (MB)` or the `Max file open time (sec)` conditions are met.

Add Output ID: Whether to append output's ID to staging location. Defaults to toggled on.

Disk space protection: Specifies whether to `Block` (default) or `Drop` incoming events when the disk space falls below the globally defined Min free disk space amount.

Remove empty staging directories: When toggled on (the default), Cribl Edge deletes empty staging directories after moving files. This prevents the proliferation of orphaned empty directories. When enabled, exposes this additional option:

- **Staging cleanup period**: How often (in seconds) to delete empty directories when **Remove staging dirs** is enabled. Defaults to `300` seconds (every 5 minutes). Minimum configurable interval is `10` seconds; maximum is `86400` seconds (every 24 hours).
- **Directory batch size**: Specifies how many directories are scanned per batch during automatic cleanup of empty staging directories. Set between `10` and `10000`, defaults to `1000`. Higher values speed up

cleanup but increase memory usage.

Force close on shutdown: Toggle on to force all staged files to close during an orderly shutdown. This triggers immediate upload or finalization of any in-progress files, regardless of **Idle time**, **File open time**, or **File size** limits. Use this option to minimize data loss risk in ephemeral or auto-scaling environments. This could result in a higher number of small files, as files are closed before reaching their configured thresholds. Ensure your Worker Group or Fleet **Drain timeout** is set high enough to allow all staged data to be finalized before shutdown completes.

Enable dead-lettering: Toggle on to set a maximum number of retries, and to move files to a designated directory when write failures exceed that limit. This prevents data flow blockage and excessive error logging due to undeliverable files. When enabled, exposes two additional fields:

- **Dead-letter location**: Specify the storage location for undeliverable files. Defaults to `$CRIBL_HOME/state/outputs/dead-letter`.
- **Maximum retry limit**: Configure the retry limit for failed file deliveries. This setting defines how many times the system will attempt to move a file to its intended location before it is deemed undeliverable and placed in the dead-letter directory. Defaults to `20`.

Endpoint: The Google Cloud Storage service endpoint. Typically, there is no reason to change the default https://storage.googleapis.com endpoint.

Object ACL: Select an Access Control List to assign to uploaded objects. Defaults to `private`.

Storage class: Select a storage class for uploaded objects.

Signature version: Signature version to use for signing requests. Defaults to `v4`.

Reuse connections: Whether to reuse connections between requests. Toggling on (default) can improve performance.

Reject unauthorized certificates: Whether to accept certificates that cannot be verified against a valid Certificate Authority (for example, self-signed certificates). Defaults to toggled on.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Internal Fields

Cribl Edge uses a set of internal fields to assist in forwarding data to a Destination.

Field for this Destination:

- `__partition`

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

Common Issue

Nonspecific messages from Google Cloud of the form `Error: failed to close file` can indicate problems with the permissions listed above.

12.3.7. KAFKA

You can send data from Cribl Edge to the following Kafka services:

Destination	Description
Kafka	Send data to a Kafka topic
Confluent Cloud	Send data to Kafka topics on Confluent Cloud
Amazon MSK	Send data to a topic in Amazon MSK

12.3.7.1. Kafka Destination

Cribl Edge supports sending data to a Kafka topic.

Type: Streaming | TLS Support: Configurable | PQ Support: Yes

Kafka uses a binary protocol over TCP. It does not support HTTP proxies, so Cribl Edge must send events directly to receivers. You might need to adjust your firewall rules to allow this traffic.

Configure Cribl Edge to Output to Kafka

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this Kafka definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
    - **Description**: Optionally, enter a description.
    - **Bootstrap servers**: List of Kafka bootstrap servers to connect to (for example, `localhost:9092`).
    - **Topic**: The topic on which to publish events. Can be overwritten using event's `__topicOut` field.
3. Next, you can configure the following **Optional Settings**:
    - **Acknowledgments**: Select the number of required acknowledgments. Defaults to `Leader`.
    - **Record data format**: Format to use to serialize events before writing to Kafka. Defaults to `JSON`. When set to `Protobuf`, the **Protobuf Format Settings** section (left tab) becomes available.
    - **Compression**: Codec to compress the data before sending to Kafka. Select `None`, `Gzip` `Snappy`, `LZ4`, or `ZSTD`.

      Cribl strongly recommends enabling compression. Doing so improves Cribl Edge's performance, enabling faster data transfer using less bandwidth.

- **Backpressure behavior**: Select whether to block, drop, or queue incoming events when all receivers are exerting backpressure. Defaults to `Block`.
- **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

4. Optionally, you can adjust the Persistent Queue, TLS, Authentication, Processing, Retries, and Advanced settings outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

> On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.
>
> This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.

Mode	Description
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.



If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5` GB. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.



Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Calculating the Time PQ Will Take to Engage

PQ will not engage until Cribl Edge has exhausted all attempts to send events to the Kafka receiver. This can take several minutes if requests continue to fail or time out.

To calculate the longest possible time this can take, multiply the values of **Request timeout** and **Retry limit**. For the default values (`60` seconds and `5`, respectively), this would be `60` seconds times `5` retries = 300 seconds, or 5 minutes.

TLS Settings (Client Side)

Enabled Default is toggled off. When toggled on:

- **Validate server certs**: Require client to reject any connection that is not authorized by a CA in the **CA certificate path**, or by another trusted CA (such as the system's CA). Default is toggled off.

Server name (SNI): Leave this field blank. See Connecting to Kafka below.

Minimum TLS version: Optionally, select the minimum TLS version to use when connecting.

Maximum TLS version: Optionally, select the maximum TLS version to use when connecting.

Certificate: Select an existing certificate name from the drop-down or add a new certificate by selecting **Create**. See Add a TLS Certificate to a Fleet for details.

CA certificate path: Path on client containing CA certificates (in PEM format) to use to verify the server's cert. Path can reference `$ENV_VARS`.

Private key path (mutual auth): Path on client containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Certificate path (mutual auth): Path on client containing certificates in (PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Passphrase: Passphrase to use to decrypt private key.

Authentication

This section documents the SASL (Simple Authentication and Security Layer) authentication settings to use when connecting to brokers. Using TLS is highly recommended.

Enabled: Default is toggled off. When toggled on:

SASL mechanism: Use this drop-down to select the SASL authentication mechanism to use. The mechanism you select determines the controls displayed below.

PLAIN, SCRAM-256, or SCRAM-512

With any of these SASL mechanisms, select one of the following buttons:

Manual: Displays **Username** and **Password** fields to enter your Kafka credentials directly.

Secret: This option exposes a **Credentials secret** drop-down in which you can select a stored text secret that references your Kafka credentials. A **Create** link is available to store a new, reusable secret.

GSSAPI/Kerberos

Kerberos authentication is not supported for Cribl-managed Workers in Cribl.Cloud, but it is enabled for customer-managed (hybrid or on-prem) Fleets, whether the Leader is based in Cribl.Cloud or on-prem.

Selecting Kerberos as the authentication mechanism displays the following options:

Keytab location: Enter the location of the keytab file for the authentication principal.

Principal: Enter the authentication principal, for example: `kafka_user@example.com`.

Broker service class: Enter the Kerberos service class for Kafka brokers, for example: `kafka`.

You will also need to set up your environment and configure the Cribl Stream host for use with Kerberos. For details, see Kafka Authentication with Kerberos.

Schema Registry

This section governs Kafka Schema Registry Authentication for data encoded with Avro or JSON schemas, with schemas stored in the Confluent Schema Registry. Cribl Edge determines the schema type by inspecting the schema registered in the Confluent Schema Registry for the provided key and value schema IDs.

Enabled: Defaults to toggled off. When toggled on, displays the following controls:

Schema registry URL: URL for access to the Confluent Schema Registry. (For example, `http://localhost:8081`.)

Default key schema ID: Used when `__keySchemaIdOut` is not present to transform key values. Leave blank if key transformation is not required by default.

Default value schema ID: Used when `__valueSchemaIdOut` not present to transform `_raw`. Leave blank if value transformation is not required by default.

Connection timeout (ms): Specifies the maximum time allowed for establishing a successful connection to the schema registry. Defaults to 30000 ms (30 seconds). Valid range is 1000 ms - 60000 ms (or 1 sec to 60 sec).

Request timeout (ms): Maximum time to wait for a successful request from the schema registry. Defaults to 60000 ms (or 1 minute).

Retry Limit: Max number of times to retry requests to the schema registry before dropping the outgoing message. Defaults to 1; enter 0 to not retry at all.

Authentication enabled: Toggle on if your schema registry requires authentication.

- **Credentials secret**: Select or **Create** a stored text secret containing the username and password provided by your schema registry administrator.

TLS enabled: defaults to toggled off. When toggled on, displays the following TLS settings for the Schema Registry (in the same format as the TLS Settings (Client Side) above):

- **Validate server certs**: Require client to reject any connection that is not authorized by a CA in the **CA certificate path**, or by another trusted CA (such as the system's CA). Default is toggled off.
- **Server name (SNI)**: Server name for the SNI (Server Name Indication) TLS extension. This must be a host name, not an IP address.
- **Minimum TLS version**: Optionally, select the minimum TLS version to use when connecting.
- **Maximum TLS version**: Optionally, select the maximum TLS version to use when connecting.
- **Certificate**: Select an existing certificate name from the drop-down or add a new certificate by selecting **Create**. See Add a TLS Certificate to a Fleet for details.
- **CA certificate path**: Path on client containing CA certificates (in PEM format) to use to verify the server's cert. Path can reference `$ENV_VARS`.
- **Private key path (mutual auth)**: Path on client containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

- **Certificate path (mutual auth)**: Path on client containing certificates in (PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.
- **Passphrase**: Passphrase to use to decrypt private key.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Retries

These settings provide flexibility in handling retries for failed messages, allowing you to balance between quick retries and avoiding excessive load on the system.

The default configuration starts with a 300 milliseconds retry interval, doubles the interval after each retry, and caps the maximum retry interval at 30 seconds. The system will attempt to retry the request up to five times before considering it a failure.

Retry limit: Maximum number of times to retry a failed request before the message fails. Defaults to `5`. Enter `0` to not retry at all. For example, if set to `5`, the system will attempt to retry the request up to five times before considering it a failure.

Initial retry interval (ms): Initial value used to calculate the retry interval, in milliseconds. This value determines the starting point for the retry delay. The default (and minimum) is `300` ms (three seconds). The maximum allowed value is `600000` ms (10 minutes). For example, if set to `1000` ms, the first retry will occur after one second.

Backoff multiplier: Set the backoff multiplier (`2`–`20`) to control the retry frequency for failed messages. The multiplier is used in an exponential backoff formula to increase the delay between retries. For faster retries, use a lower multiplier. For slower retries with more delay between attempts, use a higher multiplier. For example, with an initial retry interval of `1000` ms and a multiplier of `2`, the retry intervals will be 1,000 ms, 2,000 ms, 4,000 ms, and so on. See the Kafka documentation for details.

Backoff limit (ms): The maximum wait time for a retry, in milliseconds. This setting caps the exponential backoff delay to prevent excessively long wait times. The default (and minimum) value is `30000` ms (30 seconds), and the maximum is `180000` ms (180 seconds). For example, if the calculated retry interval exceeds 180,000 ms, the retry will occur after 180,000 ms instead.

Advanced Settings

Record size limit (KB, uncompressed): Maximum size (KB) of each record batch before compression. Setting should be < `message.max.bytes settings` in Kafka brokers. Defaults to `768`.

Events-per-batch limit: Maximum number of events in a batch before forcing a flush. Defaults to `1000`.

Flush period (sec): Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Connection timeout (ms): Maximum time to wait for a successful connection. Defaults to `10000` ms (10 seconds). Valid range is `1000` to `3600000` ms (1 second to 1 hour).

Request timeout (ms): Maximum time to wait for a successful request. Defaults to `60000` ms (1 minute).

Authentication timeout (ms): Maximum time to wait for Kafka to respond to an authentication request. Defaults to `1000` (1 second).

Reauthentication threshold (ms): If the broker requires periodic reauthentication, this setting defines how long before the reauthentication timeout Cribl Edge initiates the reauthentication. Defaults to `10000` (10 seconds).

A small value for this setting, combined with high network latency, might prevent the Destination from reauthenticating before the Kafka broker closes the connection.

A large value might cause the Destination to send reauthentication messages too soon, wasting bandwidth.

The Kafka setting `connections.max.reauth.ms` controls the reuthentication threshold on the Kafka side.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Protobuf Format Settings

Definition set: From the drop-down, select `OpenTelemetry`. This makes the **Object type** setting available.

Object type: From the drop-down, select `Logs`, `Metrics`, or `Traces`.

Working with Protobufs

This Destination supports supports Binary Protobuf payload encoding. The Protobufs it sends can encode traces, metrics, or logs, the three types of telemetry data defined in the OpenTelemetry Project's Signals documentation:

- A **trace** tracks the progression of a single request.
    - Each trace is a tree of **spans**.
    - A span object represents the work being done by the individual services, or components, involved in a request as that request flows through a system.
- A **metric** provides aggreggated statistical information.
    - A metric contains individual measurements called **data points**.
- A **log**, in OpenTelemetry terms, is "any data that is not part of a distributed trace or a metric".

When configuring Pipelines (including pre-processing and post-processing Pipelines), you need to ensure that events sent to this Destination conform to the relevant Protobuf specification:

- For traces, opentelemetry/proto/trace/v1/trace.proto.
- For metrics, opentelemetry/proto/metrics/v1/metrics.proto.
- For logs, opentelemetry/proto/logs/v1/logs.proto.

This Destination will drop non-conforming events. If the Destination encounters a parsing error when trying to convert an event to OTLP, it discards the event and Cribl Edge logs the error.

Connecting to Kafka

> **Leave the TLS Settings > Server name (SNI) field blank**
>
> In Cribl Edge's Kafka-based Sources and Destinations (including this one), the Kafka library that Cribl Edge uses manages SNI (Server Name Indication) without any input from Cribl Edge. Therefore, you should leave the **TLS Settings** > **Server name (SNI)** field blank.
>
> Setting this field in the Cribl Edge UI can cause traffic to be routed to the wrong brokers, because it interferes with the Kafka library's operation.

Connecting to a Kafka cluster involves using hostnames for two key components: bootstrap servers and brokers.

Brokers are servers that comprise the storage layer in a Kafka cluster. Bootstrap servers handle the initial connection to the Kafka cluster, and then return the list of brokers. A broker list can run into the hundreds. Every Kafka cluster has a `bootstrap.servers` property, defined as either a single `hostname:port` K-V pair, or a list of them. If Cribl Edge tries to connect via one bootstrap server and that fails, Cribl Edge then tries another one on the list.

In the **General Settings** > **Bootstrap servers** list, you can enter either the hostnames of brokers that your Kafka server has been configured to use, or, the hostnames of one or more bootstrap servers. If Kafka returns a list of brokers that's longer than the list you entered, Cribl Edge keeps the full list internally. Cribl Edge neither saves the list nor makes it available in the UI. The connection process simply starts at the beginning whenever the Source or Destination is started.

Here's an overview of the connection process:

1. From the **General Settings** > **Bootstrap servers** list - where each broker is listed as a hostname and port - Cribl Edge takes a hostname and resolves it to an IP address.
2. Cribl Edge makes a connection to that IP address. Notwithstanding the fact that Cribl Edge resolved **one** particular hostname to that IP address, there may be **many** services running at that IP address - each with its own distinct hostname.
3. Cribl Edge establishes TLS security for the connection.

Although SNI is managed by the Kafka library rather than in the Cribl Edge UI, you might want to know how it fits into the connection process. The purpose of the SNI is to specify one hostname - that is, service - among many that might be running on a given IP address within a Kafka cluster. Excluding the other services is one way that TLS makes the connection more secure.

Working with Event Timestamps

By default, when an incoming event reaches this Destination, the underlying Kafka JS library adds a `timestamp` field set to the current time at that moment. The library gets the current time from the `Date.now()` JavaScript method. Events that this Destination sends to downstream Kafka receivers include this `timestamp` field.

Some production situations require the `timestamp` value to be the time an incoming event was originally created by an upstream sender, not the current time of its arrival at this Destination. For example, suppose the events were originally created over a wide time range, and arrive at the Destination hours or days later. In this case, the original creation time might be important to retain in events Cribl Edge sends to downstream Kafka receivers.

Here is how to satisfy such a requirement:

1. In a Pipeline that routes events to this Destination, use an Eval Function to add a field named `__kafkaTime` and write the incoming event's original creation time into that field. The value of `__kafkaTime` **must** be in Unix time (either seconds or milliseconds - both will work). For context, see this explanation of the related fields `_time` and `__origTime`.
2. This Destination will recognize the `__kafkaTime` field and write its value into the `timestamp` field. (This is similar to the way you can use the `__topicOut` field to overwrite a topic setting, as described above.)

If the `__kafkaTime` field is **not** present, the Destination will apply the default behavior (using `Date.now()`) described previously.

Internal Fields

Cribl Edge uses a set of internal fields to assist in forwarding data to a Destination.

Fields for this Destination:

- `__topicOut`
- `__key`
- `__headers`
- `__keySchemaIdOut`
- `__valueSchemaIdOut`

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

12.3.7.2. Confluent Cloud Destination

Cribl Edge supports sending data to Kafka topics on the Confluent Cloud managed Kafka platform.

Type: Streaming | TLS Support: Configurable | PQ Support: Yes

Confluent Cloud uses a binary protocol over TCP. It does not support HTTP proxies, so Cribl Edge must send events directly to receivers. You might need to adjust your firewall rules to allow this traffic.

Send Kafka Topic Data to Confluent Cloud

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this Destination definition.
    - **Description**: Optionally, enter a description.
    - **Bootstrap servers**: List of Confluent Cloud bootstrap servers to connect to (for example, `myAccount.confluent.cloud:9092`).
    - **Topic**: The topic on which to publish events. Can be overwritten using event's `__topicOut` field.
3. Next, you can configure the following **Optional Settings**:
    - **Acknowledgments**: Select the number of required acknowledgments. Defaults to `Leader`.
    - **Record data format**: Format to use to serialize events before writing to Kafka. Defaults to `JSON`. When set to `Protobuf`, the **Protobuf Format Settings** section (left tab) becomes available.
    - **Compression**: Codec to use to compress the data before sending to Kafka. Select `None`, `Gzip`, `Snappy`, `LZ4`, or `ZSTD`. Defaults to `Gzip`.
    - **Backpressure behavior**: Select whether to block, drop, or queue incoming events when all receivers are exerting backpressure. Defaults to `Block`.

- **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

4. Optionally, configure any Persistent Queue, Authentication, TLS, Schema Registry, Processing, Retries, and Advanced settings outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5 GB`. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.



Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Calculating the Time PQ Will Take to Engage

PQ will not engage until Cribl Edge has exhausted all attempts to send events to the Kafka receiver. This can take several minutes if requests continue to fail or time out.

To calculate the longest possible time this can take, multiply the values of **Request timeout** and **Retry limit**. For the default values (`60` seconds and `5`, respectively), this would be `60` seconds times `5` retries = 300 seconds, or 5 minutes.

TLS Settings (Client Side)

Enabled When toggled on (default):

Autofill?: This setting is experimental.

Validate server certs: Toggle on to reject certificates that are **not** authorized by a CA in the **CA certificate path**, nor by another trusted CA (such as the system's CA).

Server name (SNI): Leave this field blank. See Connecting to Kafka below.

Minimum TLS version: Optionally, select the minimum TLS version to use when connecting.

Maximum TLS version: Optionally, select the maximum TLS version to use when connecting.

Certificate: Select an existing certificate name from the drop-down or add a new certificate by selecting **Create**. See Add a TLS Certificate to a Fleet for details.

CA certificate path: Path on client containing CA certificates (in PEM format) to use to verify the server's cert. Path can reference `$ENV_VARS`.

Private key path (mutual auth): Path on client containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Certificate path (mutual auth): Path on client containing certificates in (PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Passphrase: Passphrase to use to decrypt private key.

Authentication

This section documents the SASL (Simple Authentication and Security Layer) authentication settings to use when connecting to bootstrap servers. Using TLS is highly recommended.

Enabled: Default is toggled off. When toggled on:

- **SASL mechanism**: Select the SASL (Simple Authentication and Security Layer) authentication mechanism to use. Defaults to `PLAIN`. `SCRAM-SHA-256`, `SCRAM-SHA-512`, and `GSSAPI/Kerberos` are also available. The mechanism you select determines the controls displayed below.

PLAIN, SCRAM-256, or SCRAM-512

With any of these SASL mechanisms, select one of the following buttons:

Manual: Displays **API key** and **Secret** fields to enter your Kafka credentials directly.

Secret: This option exposes a **Credentials secret** drop-down in which you can select a stored text secret that references your Kafka credentials. A **Create** link is available to store a new, reusable secret.

GSSAPI/Kerberos

Kerberos authentication is not supported for Cribl-managed Workers in Cribl.Cloud, but it is enabled for on-prem Fleets, whether the Leader is based in Cribl.Cloud or on-prem.

Selecting Kerberos as the authentication mechanism displays the following options:

Keytab location: Enter the location of the keytab file for the authentication principal.

Principal: Enter the authentication principal, for example: `kafka_user@example.com`.

Broker service class: Enter the Kerberos service class for Kafka brokers, for example: `kafka`.

You will also need to set up your environment and configure the Cribl Stream host for use with Kerberos. See Kafka Authentication with Kerberos for further detail.

Schema Registry

This section governs Kafka Schema Registry Authentication for data encoded with Avro or JSON schemas, with schemas stored in the Confluent Schema Registry. Cribl Edge determines the schema type by inspecting the schema registered in the Confluent Schema Registry for the provided key and value schema IDs.

Enabled: Default is toggled off. When toggled on, displays the following controls:

Schema registry URL: URL for access to the Confluent Schema Registry. (For example, `http://localhost:8081`.)

Default key schema ID: Used when `__keySchemaIdOut` is not present to transform key values. Leave blank if key transformation is not required by default.

Default value schema ID: Used when `__valueSchemaIdOut` not present to transform `_raw`. Leave blank if value transformation is not required by default.

Connection timeout (ms): Specifies the maximum time allowed for establishing a successful connection to the schema registry. Defaults to 30000 ms (30 seconds). Valid range is 1000 ms to 60000 ms (or 1 sec to 60 sec).

Request timeout (ms): Maximum time to wait for a successful request from the schema registry. Defaults to 60000 ms (or 1 minute).

Retry Limit: Max number of times to retry requests to the schema registry before dropping the outgoing message. Defaults to 1; enter 0 to not retry at all.

Authentication enabled: Toggle on if your schema registry requires authentication.

- **Credentials secret**: Select or **Create** a stored text secret containing the username and password provided by your schema registry administrator.

TLS enabled: defaults to toggled off. When toggled on, displays the following TLS settings for the Schema Registry (in the same format as the TLS Settings (Client Side) above):

- **Validate server certs**: Require client to reject any connection that is not authorized by a CA in the **CA certificate path**, or by another trusted CA (such as the system's CA). Default is toggled off.
- **Server name (SNI)**: Server name for the SNI (Server Name Indication) TLS extension. This must be a host name, not an IP address.



With a dedicated Confluent Cloud cluster hosted in Microsoft Azure, be sure to specify the **Server name (SNI)**. If this is omitted, Confluent Cloud might reset the connection to Cribl Edge.

- **Minimum TLS version**: Optionally, select the minimum TLS version to use when connecting.
- **Maximum TLS version**: Optionally, select the maximum TLS version to use when connecting.
- **Certificate**: Select an existing certificate name from the drop-down or add a new certificate by selecting **Create**. See Add a TLS Certificate to a Fleet for details.
- **CA certificate path**: Path on client containing CA certificates (in PEM format) to use to verify the server's cert. Path can reference `$ENV_VARS`.
- **Private key path (mutual auth)**: Path on client containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

- **Certificate path (mutual auth)**: Path on client containing certificates in (PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.
- **Passphrase**: Passphrase to use to decrypt private key.

Processing Settings Post-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline to process data before it is sent through this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Retries

These settings provide flexibility in handling retries for failed messages, allowing you to balance between quick retries and avoiding excessive load on the system.

The default configuration starts with a 300 milliseconds retry interval, doubles the interval after each retry, and caps the maximum retry interval at 30 seconds. The system will attempt to retry the request up to five times before considering it a failure.

Retry limit: Maximum number of times to retry a failed request before the message fails. Defaults to `5`. Enter `0` to not retry at all. For example, if set to `5`, the system will attempt to retry the request up to five times before considering it a failure.

Initial retry interval (ms): Initial value used to calculate the retry interval, in milliseconds. This value determines the starting point for the retry delay. The default (and minimum) is `300` ms (three seconds). The maximum allowed value is `600000` ms (10 minutes). For example, if set to `1000` ms, the first retry will occur after one second.

Backoff multiplier: Set the backoff multiplier (`2`–`20`) to control the retry frequency for failed messages. The multiplier is used in an exponential backoff formula to increase the delay between retries. For faster retries, use a lower multiplier. For slower retries with more delay between attempts, use a higher multiplier. For example, with an initial retry interval of `1000` ms and a multiplier of `2`, the retry intervals will be 1,000 ms, 2,000 ms, 4,000 ms, and so on. See the Kafka documentation for details.

Backoff limit (ms): The maximum wait time for a retry, in milliseconds. This setting caps the exponential backoff delay to prevent excessively long wait times. The default (and minimum) value is `30000` ms (30 seconds), and the maximum is `180000` ms (180 seconds). For example, if the calculated retry interval exceeds 180,000 ms, the retry will occur after 180,000 ms instead.

Advanced Settings

Record size limit (KB, uncompressed): Maximum size (KB) of each record batch before compression. Setting should be < `message.max.bytes settings` in Kafka brokers. Defaults to `768`.

Events-per-batch limit: Maximum number of events in a batch before forcing a flush. Defaults to `1000`.

Flush period (sec): Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Connection timeout (ms): Maximum time to wait for a successful connection. Defaults to `10000` ms (10 seconds). Valid range is `1000` to `3600000` ms (1 second to 1 hour).

Request timeout (ms): Maximum time to wait for a successful request. Defaults to `60000` ms (1 minute).

Authentication timeout (ms): Maximum time to wait for Kafka to respond to an authentication request. Defaults to `1000` (1 second).

Reauthentication threshold (ms): If the broker requires periodic reauthentication, this setting defines how long before the reauthentication timeout Cribl Edge initiates the reauthentication. Defaults to `10000` (10 seconds).

A small value for this setting, combined with high network latency, might prevent the Destination from reauthenticating before the Kafka broker closes the connection.

A large value might cause the Destination to send reauthentication messages too soon, wasting bandwidth.

The Kafka setting `connections.max.reauth.ms` controls the reuthentication threshold on the Kafka side.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Protobuf Format Settings

Definition set: From the drop-down, select `OpenTelemetry`. This makes the **Object type** setting available.

Object type: From the drop-down, select `Logs`, `Metrics`, or `Traces`.

Working with Protobufs

This Destination supports supports Binary Protobuf payload encoding. The Protobufs it sends can encode traces, metrics, or logs, the three types of telemetry data defined in the OpenTelemetry Project's Signals documentation:

- A **trace** tracks the progression of a single request.
    - Each trace is a tree of **spans**.
    - A span object represents the work being done by the individual services, or components, involved in a request as that request flows through a system.
- A **metric** provides aggreggated statistical information.
    - A metric contains individual measurements called **data points**.
- A **log**, in OpenTelemetry terms, is "any data that is not part of a distributed trace or a metric".

When configuring Pipelines (including pre-processing and post-processing Pipelines), you need to ensure that events sent to this Destination conform to the relevant Protobuf specification:

- For traces, opentelemetry/proto/trace/v1/trace.proto.
- For metrics, opentelemetry/proto/metrics/v1/metrics.proto.
- For logs, opentelemetry/proto/logs/v1/logs.proto.

This Destination will drop non-conforming events. If the Destination encounters a parsing error when trying to convert an event to OTLP, it discards the event and Cribl Edge logs the error.

Connecting to Kafka

Leave the TLS Settings > Server name (SNI) field blank

In Cribl Edge's Kafka-based Sources and Destinations (including this one), the Kafka library that Cribl Edge uses manages SNI (Server Name Indication) without any input from Cribl Edge. Therefore, you should leave the **TLS Settings** > **Server name (SNI)** field blank.

Setting this field in the Cribl Edge UI can cause traffic to be routed to the wrong brokers, because it interferes with the Kafka library's operation. This is especially important for Confluent Cloud Dedicated clusters, which rely on SNI - as managed by the Kafka library - for routing.

Connecting to a Kafka cluster involves using hostnames for two key components: bootstrap servers and brokers.

Brokers are servers that comprise the storage layer in a Kafka cluster. Bootstrap servers handle the initial connection to the Kafka cluster, and then return the list of brokers. A broker list can run into the hundreds. Every Kafka cluster has a `bootstrap.servers` property, defined as either a single `hostname:port` K-V pair,

or a list of them. If Cribl Edge tries to connect via one bootstrap server and that fails, Cribl Edge then tries another one on the list.

In the **General Settings** > **Bootstrap servers** list, you can enter either the hostnames of brokers that your Kafka server has been configured to use, or, the hostnames of one or more bootstrap servers. If Kafka returns a list of brokers that's longer than the list you entered, Cribl Edge keeps the full list internally. Cribl Edge neither saves the list nor makes it available in the UI. The connection process simply starts at the beginning whenever the Source or Destination is started.

Here's an overview of the connection process:

1. From the **General Settings** > **Bootstrap servers** list - where each broker is listed as a hostname and port - Cribl Edge takes a hostname and resolves it to an IP address.
2. Cribl Edge makes a connection to that IP address. Notwithstanding the fact that Cribl Edge resolved **one** particular hostname to that IP address, there may be **many** services running at that IP address - each with its own distinct hostname.
3. Cribl Edge establishes TLS security for the connection.

Although SNI is managed by the Kafka library rather than in the Cribl Edge UI, you might want to know how it fits into the connection process. The purpose of the SNI is to specify one hostname - that is, service - among many that might be running on a given IP address within a Kafka cluster. Excluding the other services is one way that TLS makes the connection more secure.

Internal Fields

Cribl Edge uses a set of internal fields to assist in forwarding data to a Destination.

Fields for this Destination:

- `__topicOut`
- `__key`
- `__headers`
- `__keySchemaIdOut`
- `__valueSchemaIdOut`

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

12.3.7.3. Amazon MSK Destination

Cribl Edge supports sending data to an Amazon Managed Streaming for Apache Kafka (MSK) topic.

Type: Streaming | TLS Support: Configurable | PQ Support: Yes

Kafka uses a binary protocol over TCP. It does not support HTTP proxies, so Cribl Edge must send events directly to receivers. You might need to adjust your firewall rules to allow this traffic.

Configuring Cribl Edge to Output to Amazon MSK

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this Amazon MSK Destination definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
    - **Description**: Optionally, enter a description.
    - **Bootstrap servers**: List of Kafka bootstrap servers to connect to (for example, `localhost:9092`).
    - **Topic**: The topic on which to publish events. Can be overwritten using the event's `__topicOut` field.
    - **Region**: From the drop-down, select the name of the AWS Region where your Amazon MSK cluster is located.
3. Next, you can configure the following **Optional Settings**:
    - **Acknowledgments**: Select the number of required acknowledgments. Defaults to `Leader`.
    - **Record data format**: Format to use to serialize events before writing to Kafka. Defaults to `JSON`. When set to `Protobuf`, the **Protobuf Format Settings** section (left tab) becomes available.
    - **Compression**: Codec to compress the data before sending to Kafka. Select `None`, `Gzip`, `Snappy`, `LZ4`, or `ZSTD`. Defaults to `Gzip`.



Cribl strongly recommends enabling compression. Doing so improves Cribl Edge's performance, enabling faster data transfer using less bandwidth.

- **Backpressure behavior**: Select whether to block, drop, or queue incoming events when all receivers are exerting backpressure. Defaults to `Block`.
- **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

4. Optionally, you can adjust the Persistent Queue, TLS, Authentication, Processing, Retries, and Advanced settings outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.

Mode	Description
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5 GB`. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.



Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Calculating the Time PQ Will Take to Engage

PQ will not engage until Cribl Edge has exhausted all attempts to send events to the Kafka receiver. This can take several minutes if requests continue to fail or time out.

To calculate the longest possible time this can take, multiply the values of **Request timeout** and **Retry limit**. For the default values (`60` seconds and `5`, respectively), this would be `60` seconds times `5` retries = 300 seconds, or 5 minutes.

TLS Settings (Client Side)



For Amazon MSK Sources and Destinations:

- IAM is the only type of authentication that Cribl Edge supports.
- Because IAM auth requires TLS, TLS is automatically enabled.

- **Validate server certs**: Require client to reject any connection that is not authorized by a CA in the **CA certificate path**, or by another trusted CA (such as the system's CA). Defaults to toggled on.

Server name (SNI): Leave this field blank. See Connecting to Kafka below.

Certificate: Select an existing certificate name from the drop-down or add a new certificate by selecting **Create**. See Add a TLS Certificate to a Fleet for details.

CA certificate path: Path on client containing CA certificates (in PEM format) to use to verify the server's cert. Path can reference `$ENV_VARS`.

Private key path (mutual auth): Path on client containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Certificate path (mutual auth): Path on client containing certificates in (PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required.**

Passphrase: Passphrase to use to decrypt private key.

Minimum TLS version: Optionally, select the minimum TLS version to use when connecting.

Maximum TLS version: Optionally, select the maximum TLS version to use when connecting.

Authentication

Use the **Authentication method** drop-down to select an AWS authentication method.

Auto: This default option uses the AWS SDK for JavaScript to automatically obtain credentials in the following order of attempts:

1. **Environment Variables**: Loaded from environment variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`.
2. **IAM Identity Center (SSO)**: Loaded from AWS IAM Identity Center configuration.
3. **Shared Credentials File**: Loaded from the shared credentials file (`~/.aws/credentials`).
4. **IAM Roles for Amazon EC2/ECS**: Loaded from AWS Identity and Access Management (IAM) roles attached to an EC2 instance or ECS container.
5. **JSON File on Disk**: Loaded from a JSON file on disk.
6. **Other Credential-Provider Classes**: Other credential-provider classes provided by the AWS SDK for JavaScript.

The `Auto` method works both when running on AWS and in other environments where the necessary credentials are available through one of the above methods.



SSO Providers

When using the auto authentication method, you can leverage SSO providers like SAML and Okta to issue temporary credentials. These credentials should be set in the environment variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`. The AWS SDK will then use these environment variables to authenticate.

Manual: If not running on AWS, you can select this option to enter a static set of user-associated IAM credentials (your access key and secret key) directly or by reference. This is useful for Workers not in an AWS VPC, such as those running in a private cloud. The **Manual** option exposes these corresponding additional fields:

- **Access key**: Enter your AWS access key. If not present, will fall back to the `env.AWS_ACCESS_KEY_ID` environment variable, or to the metadata endpoint for IAM role credentials.

- **Secret key**: Enter your AWS secret key. If not present, will fall back to the `env.AWS_SECRET_ACCESS_KEY` environment variable, or to the metadata endpoint for IAM credentials.

Secret: If not running on AWS, you can select this option to supply a stored secret that references an AWS access key and secret key. The **Secret** option exposes this additional field:

- **Secret key pair**: Use the drop-down to select an API key/secret key pair that you've configured in Cribl Edge's secrets manager. A **Create** link is available to store a new, reusable secret.

Assume Role

When using **Assume Role** to access resources in a different Region than Cribl Edge, you can target the AWS Security Token Service (STS) endpoint specific to that Region by using the `CRIBL_AWS_STS_REGION` environment variable on your Edge Node. To specify a custom STS endpoint, set the `CRIBL_AWS_STS_ENDPOINT` environment variable. If both are set, `CRIBL_AWS_STS_ENDPOINT` takes precedence. Setting an invalid Region or endpoint results in a fallback to the global STS endpoint.

Enable for MSK: Toggle on to use Assume Role credentials to access MSK.

AssumeRole ARN: Enter the Amazon Resource Name (ARN) of the role to assume.

External ID: Enter the External ID to use when assuming role. This is required only when assuming a role that requires this ID in order to delegate third-party access. For details, see AWS' documentation.

Duration (seconds): Duration of the Assumed Role's session, in seconds. Minimum is 900 (15 minutes). Maximum is 43200 (12 hours). Defaults to 3600 (1 hour).

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Retries

These settings provide flexibility in handling retries for failed messages, allowing you to balance between quick retries and avoiding excessive load on the system.

The default configuration starts with a 300 milliseconds retry interval, doubles the interval after each retry, and caps the maximum retry interval at 30 seconds. The system will attempt to retry the request up to five times before considering it a failure.

Retry limit: Maximum number of times to retry a failed request before the message fails. Defaults to `5`. Enter `0` to not retry at all. For example, if set to `5`, the system will attempt to retry the request up to five times before considering it a failure.

Initial retry interval (ms): Initial value used to calculate the retry interval, in milliseconds. This value determines the starting point for the retry delay. The default (and minimum) is `300` ms (three seconds). The maximum allowed value is `600000` ms (10 minutes). For example, if set to `1000` ms, the first retry will occur after one second.

Backoff multiplier: Set the backoff multiplier (`2–20`) to control the retry frequency for failed messages. The multiplier is used in an exponential backoff formula to increase the delay between retries. For faster retries, use a lower multiplier. For slower retries with more delay between attempts, use a higher multiplier. For example, with an initial retry interval of `1000` ms and a multiplier of `2`, the retry intervals will be 1,000 ms, 2,000 ms, 4,000 ms, and so on. See the Kafka documentation for details.

Backoff limit (ms): The maximum wait time for a retry, in milliseconds. This setting caps the exponential backoff delay to prevent excessively long wait times. The default (and minimum) value is `30000` ms (30 seconds), and the maximum is `180000` ms (180 seconds). For example, if the calculated retry interval exceeds 180,000 ms, the retry will occur after 180,000 ms instead.

Advanced Settings

Record size limit (KB, uncompressed): Maximum size (KB) of each record batch before compression. Setting should be < `message.max.bytes settings` in Kafka brokers. Defaults to `768`.

Events-per-batch limit: Maximum number of events in a batch before forcing a flush. Defaults to `1000`.

Flush period (sec): Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Connection timeout (ms): Maximum time to wait for a connection to complete successfully. Defaults to `10000` ms (10 seconds). Valid range is `1000` to `3600000` ms (1 second to 1 hour).

Request timeout (ms): Maximum time to wait for Kafka to respond to a request. Defaults to `60000` ms (1 minute).

Authentication timeout (ms): Maximum time to wait for Kafka to respond to an authentication request. Defaults to `1000` (1 second).

Reauthentication threshold (ms): If the broker requires periodic reauthentication, this setting defines how long before the reauthentication timeout Cribl Edge initiates the reauthentication. Defaults to `10000` (10 seconds).

A small value for this setting, combined with high network latency, might prevent the Destination from reauthenticating before the Kafka broker closes the connection.

A large value might cause the Destination to send reauthentication messages too soon, wasting bandwidth.

The Kafka setting `connections.max.reauth.ms` controls the reuthentication threshold on the Kafka side.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Protobuf Format Settings

Definition set: From the drop-down, select `OpenTelemetry`. This makes the **Object type** setting available.

Object type: From the drop-down, select `Logs`, `Metrics`, or `Traces`.

Working with Protobufs

This Destination supports supports Binary Protobuf payload encoding. The Protobufs it sends can encode traces, metrics, or logs, the three types of telemetry data defined in the OpenTelemetry Project's Signals documentation:

- A **trace** tracks the progression of a single request.
    - Each trace is a tree of **spans**.
    - A span object represents the work being done by the individual services, or components, involved in a request as that request flows through a system.
- A **metric** provides aggreggated statistical information.
    - A metric contains individual measurements called **data points**.
- A **log**, in OpenTelemetry terms, is "any data that is not part of a distributed trace or a metric".

When configuring Pipelines (including pre-processing and post-processing Pipelines), you need to ensure that events sent to this Destination conform to the relevant Protobuf specification:

- For traces, opentelemetry/proto/trace/v1/trace.proto.
- For metrics, opentelemetry/proto/metrics/v1/metrics.proto.
- For logs, opentelemetry/proto/logs/v1/logs.proto.

This Destination will drop non-conforming events. If the Destination encounters a parsing error when trying to convert an event to OTLP, it discards the event and Cribl Edge logs the error.

Connecting to Kafka



Leave the TLS Settings > Server name (SNI) field blank

In Cribl Edge's Kafka-based Sources and Destinations (including this one), the Kafka library that Cribl Edge uses manages SNI (Server Name Indication) without any input from Cribl Edge. Therefore, you should leave the **TLS Settings** > **Server name (SNI)** field blank.

Setting this field in the Cribl Edge UI can cause traffic to be routed to the wrong brokers, because it interferes with the Kafka library's operation.

Connecting to a Kafka cluster involves using hostnames for two key components: bootstrap servers and brokers.

Brokers are servers that comprise the storage layer in a Kafka cluster. Bootstrap servers handle the initial connection to the Kafka cluster, and then return the list of brokers. A broker list can run into the hundreds. Every Kafka cluster has a `bootstrap.servers` property, defined as either a single `hostname:port` K-V pair, or a list of them. If Cribl Edge tries to connect via one bootstrap server and that fails, Cribl Edge then tries another one on the list.

In the **General Settings** > **Bootstrap servers** list, you can enter either the hostnames of brokers that your Kafka server has been configured to use, or, the hostnames of one or more bootstrap servers. If Kafka returns a list of brokers that's longer than the list you entered, Cribl Edge keeps the full list internally. Cribl Edge neither saves the list nor makes it available in the UI. The connection process simply starts at the beginning whenever the Source or Destination is started.

Here's an overview of the connection process:

1. From the **General Settings** > **Bootstrap servers** list - where each broker is listed as a hostname and port - Cribl Edge takes a hostname and resolves it to an IP address.
2. Cribl Edge makes a connection to that IP address. Notwithstanding the fact that Cribl Edge resolved **one** particular hostname to that IP address, there may be **many** services running at that IP address - each with its own distinct hostname.
3. Cribl Edge establishes TLS security for the connection.

Although SNI is managed by the Kafka library rather than in the Cribl Edge UI, you might want to know how it fits into the connection process. The purpose of the SNI is to specify one hostname - that is, service - among many that might be running on a given IP address within a Kafka cluster. Excluding the other services is one way that TLS makes the connection more secure.

Working with Event Timestamps

By default, when an incoming event reaches this Destination, the underlying Kafka JS library adds a `timestamp` field set to the current time at that moment. The library gets the current time from the `Date.now()` JavaScript method. Events that this Destination sends to downstream Kafka receivers include this `timestamp` field.

Some production situations require the `timestamp` value to be the time an incoming event was originally created by an upstream sender, not the current time of its arrival at this Destination. For example, suppose the events were originally created over a wide time range, and arrive at the Destination hours or days later. In this case, the original creation time might be important to retain in events Cribl Edge sends to downstream Kafka receivers.

Here is how to satisfy such a requirement:

1. In a Pipeline that routes events to this Destination, use an Eval Function to add a field named `__kafkaTime` and write the incoming event's original creation time into that field. The value of `__kafkaTime` **must** be in Unix time (either seconds or milliseconds - both will work). For context, see this explanation of the related fields `_time` and `__origTime`.
2. This Destination will recognize the `__kafkaTime` field and write its value into the `timestamp` field. (This is similar to the way you can use the `__topicOut` field to overwrite a topic setting, as described above.)

If the `__kafkaTime` field is **not** present, the Destination will apply the default behavior (using `Date.now()`) described previously.

Internal Fields

Cribl Edge uses a set of internal fields to assist in forwarding data to a Destination.

Fields for this Destination:

- `__topicOut`
- `__key`
- `__headers`
- `__keySchemaIdOut`
- `__valueSchemaIdOut`

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

12.3.8. Metrics

You can send data from Cribl Edge to the following Metrics services:

Destination	Description
Graphite	Send data to a Graphite Destination
StatsD	Send data to a StatsD Destination
StatsD Extended	Send out data in expanded StatsD format

12.3.8.1. Graphite Destination

Cribl Edge supports sending data to a Graphite backend Destination.



Type: Streaming | TLS Support: No | PQ Support: Yes

Configure Cribl Edge to Output to a Graphite Backend

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this Graphite definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
    - **Description**: Optionally, enter a description.
    - **Destination protocol**: Protocol to use when communicating with the Destination. Defaults to `UDP`.
    - **Host**: The hostname of the Destination.
    - **Port**: Destination port. Defaults to `8125`.
3. Next, you can configure the following **Optional Settings**:
    - **Throttling**: Displayed only when **General Settings** > **Destination protocol** is set to `TCP`. Rate (in bytes per second) at which at which to throttle while writing to an output. Also takes numerical values in multiples of bytes (KB, MB, GB, etc.). Default value of `0` indicates no throttling.
    - **Backpressure behavior**: Displayed only when **General Settings** > **Destination protocol** is set to `TCP`. Select whether to block, drop, or queue events when all receivers are exerting backpressure. (Causes might include a broken or denied connection, or a rate limiter.) Defaults to `Block`.
    - **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

4. Optionally, you can adjust the Persistent Queue, Timeout, Processing, and Advanced settings outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.



If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5` GB. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.

Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Timeout Settings

This section is displayed only when **General Settings** > **Destination protocol** is set to `TCP`.

Connection timeout: Amount of time (in milliseconds) to wait for the connection to establish, before retrying. Defaults to `10000`.

Write timeout: Amount of time (milliseconds) to wait for a write to complete, before assuming connection is dead. Defaults to `60000`.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Advanced

Record size limit (bytes): Used when Protocol is UDP. Specifies the maximum size of packets sent to the Destination. (Also known as the MTU - maximum transmission unit - for the network path to the destination system.) Defaults to `512`.

Flush period (sec): Used when Protocol is TCP. Specifies how often buffers should be flushed, sending records to the Destination. Defaults to `1`.

DNS resolution period (sec): Specify the interval (in seconds) to re-resolve hostnames. This reduces the frequency of DNS lookups, improving performance. Use this setting to balance the overhead of DNS lookup

calls with the expected frequency of changes in the DNS records. Defaults to `0` seconds, meaning DNS lookups occur for every outgoing batch.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

12.3.8.2. StatsD Destination

Cribl Edge supports sending data to a StatsD Destination.



Type: Streaming | TLS Support: No | PQ Support: Yes

Configure Cribl Edge to Output via StatsD

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this StatsD definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
    - **Description**: Optionally, enter a description.
    - **Destination protocol**: Protocol to use when communicating with the Destination. Defaults to `UDP`.
    - **Host**: The hostname of the Destination.
    - **Port**: Destination port. Defaults to `8125`.
3. Next, you can configure the following **Optional Settings**:
    - **Throttling**: Displayed only when **General Settings** > **Destination protocol** is set to `TCP`. Rate (in bytes per second) at which at which to throttle while writing to an output. Also takes numerical values in multiples of bytes (KB, MB, GB, etc.). Default value of `0` indicates no throttling.
    - **Backpressure behavior**: Displayed only when **General Settings** > **Destination protocol** is set to `TCP`. Select whether to block, drop, or queue events when all receivers are exerting backpressure. (Causes might include a broken or denied connection, or a rate limiter.) Defaults to `Block`.
    - **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

4. Optionally, you can adjust the Persistent Queue, Timeout, Processing, and Advanced settings outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

> On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.
>
> This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

> If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5` GB. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.

Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Timeout Settings

This section is displayed only when **General Settings** > **Destination protocol** is set to `TCP`.

Connection timeout: Amount of time (in milliseconds) to wait for the connection to establish, before retrying. Defaults to `10000`.

Write timeout: Amount of time (milliseconds) to wait for a write to complete, before assuming connection is dead. Defaults to `60000`.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Advanced Settings

Record size limit (bytes): Used when Protocol is UDP. Specifies the maximum size of packets sent to the Destination. (Also known as the MTU - maximum transmission unit - for the network path to the Destination system.) Defaults to `512`.

Flush period (sec): Used when Protocol is TCP. Specifies how often buffers should be flushed, sending records to the Destination. Defaults to `1`.

DNS resolution period (sec): Specify the interval (in seconds) to re-resolve hostnames. This reduces the frequency of DNS lookups, improving performance. Use this setting to balance the overhead of DNS lookup

calls with the expected frequency of changes in the DNS records. Defaults to `0` seconds, meaning DNS lookups occur for every outgoing batch.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

12.3.8.3. StatsD Extended Destination

Cribl Edge's StatsD Extended Destination supports sending out data in expanded StatsD format.

The output is an expanded StatsD metric protocol that supports dimensions, along with a sample rate for counter metrics. As with StatsD, downstream components listen for application metrics over UDP or TCP, can aggregate and summarize those metrics, and can relay them to virtually any graphing or monitoring backend.

For details about the syntax expected by one common downstream service, see Splunk's Expanded StatsD Metric Protocol documentation.



Type: Streaming | TLS Support: No | PQ Support: Yes

Configure Cribl Edge to Output via StatsD Extended

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this StatsD Extended definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
    - **Description**: Optionally, enter a description.
    - **Destination protocol**: Protocol to use when communicating with the Destination. Defaults to `UDP`.
    - **Host**: The hostname of the Destination.
    - **Port**: Destination port. Defaults to `8125`.
3. Next, you can configure the following **Optional Settings**:
    - **Throttling**: Displayed only when **General Settings** > **Destination protocol** is set to `TCP`. Rate (in bytes per second) at which at which to throttle while writing to an output. Also takes numerical values in multiples of bytes (KB, MB, GB, etc.). Default value of `0` indicates no throttling.

- **Backpressure behavior**: Displayed only when **General Settings** > **Destination protocol** is set to `TCP`. Select whether to block, drop, or queue events when all receivers are exerting backpressure. (Causes might include a broken or denied connection, or a rate limiter.) Defaults to `Block`.
- **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

4. Optionally, you can adjust the Persistent Queue, Timeout, Processing, and Advanced settings outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

> On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.
>
> This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.

Mode	Description
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5` GB. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.



Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Advanced Settings

Record size limit (bytes): Used when Protocol is UDP. Specifies the maximum size of packets sent to the Destination. (Also known as the MTU - maximum transmission unit - for the network path to the Destination system.) Defaults to `512`.

Flush period (sec): Used when Protocol is TCP. Specifies how often buffers should be flushed, sending records to the Destination. Defaults to `1`.

DNS resolution period (sec): Specify the interval (in seconds) to re-resolve hostnames. This reduces the frequency of DNS lookups, improving performance. Use this setting to balance the overhead of DNS lookup calls with the expected frequency of changes in the DNS records. Defaults to `0` seconds, meaning DNS lookups occur for every outgoing batch.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

12.3.9. New Relic Ingest

You can send data from Cribl Edge to the following New Relic Ingest services:

Destination	Description
New Relic Events	Send events to New Relic via the New Relic Event API
New Relic Logs & Metrics	Send events to New Relic Log API and Metric API

12.3.9.1. New Relic Events Destination

Cribl Edge supports sending events to New Relic via the New Relic Event API. Use this Destination to export ad hoc events that New Relic ingestion treats as custom events.

To export structured log and/or metric events, use Cribl Edge's New Relic Logs & Metrics Destination.

Type: Streaming | TLS Support: Yes | PQ Support: Yes

Configure Cribl Edge to Output Events to New Relic

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this Destination.
    - **Description**: Optionally, enter a description.
    - **Region**: Select which New Relic region endpoint to use. Select `Custom` to manually enter your New Relic region's URL if it's not listed.
    - **Account ID**: Enter your New Relic account ID. (You can access this ID from New Relic's **account** drop-down, by selecting **Manage your plan**.)
    - **Event type**: Default `eventType` to apply when not specified in an event. You can use arbitrary values, as long as they do not conflict with New Relic reserved words. (Where an `eventType` is specified in an event, it will override this value.)
3. Under **Authentication**, select an **Authentication method** from the dropdown:

If an incoming event contains an internal field named `__newRelic_apiKey`, the New Relic Events Destination will use that field's value as the API key when sending the event to New Relic.

> For events that do not contain a `__newRelic_apiKey` field, the Destination will use whatever API key you have configured in this section.

- **Manual**: This default option exposes an **API key** field. Directly enter your New Relic Ingest License API key, as you created or accessed it from New Relic's **account** drop-down. (For details, see the New Relic API Keys documentation.)
- **Secret**: This option exposes an **API key (text secret)** drop-down, in which you can select a stored secret that references a New Relic Ingest License API key. A **Create** link is available to store a new, reusable secret.

4. Next, you can configure the following **Optional Settings**:
   - **Backpressure behavior**: Select whether to block, drop, or queue events when all receivers are exerting backpressure. (Causes might include a broken or denied connection, or a rate limiter.) Defaults to `Block`. For the `Persistent Queue` option, see the section just below.
   - **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page.. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
5. Optionally, you can adjust the Persistent Queue, Processing, Retries, and Advanced settings outlined in the sections below.
6. Select **Save**, then **Commit & Deploy**.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

> On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.
>
> This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a

> hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

> If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5 GB`. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.



Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Processing Settings Post-Processing

Pipeline: Pipeline to process data before sending the data out via this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Retries

Honor Retry-After header: Toggle on to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream/Edge limits the delay to 180 seconds even if the `Retry-After`

header specifies a longer delay. Any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. Toggle off to ignore all `Retry-After` headers.

Default Retry Behavior: By default, Cribl Stream/Edge applies predefined retry strategies for certain failed requests to the Destination using an exponential backoff algorithm. You can override these settings by manually configuring the retry behavior for a particular error code.

To retry requests, Cribl Stream/Edge applies the following rules by default:

Status Code	Action
Any in the `1xx, 3xx` series	Drop the request
`408, 429` errors	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached
Any other `4xx` series	Drop the request
Any `5xx` series	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached

By default, any `429 (Too Many Requests)` response that is not listed in the retry settings is treated as a non-retryable `4xx` error and dropped. If you want `429` responses retried, ensure `429` is included in the **Settings for failed HTTP requests**.

Settings for failed HTTP requests: You can also manually configure Cribl Stream/Edge to automatically retry requests that receive particular HTTP response status codes.

Upon receiving a response code that's on the list, Cribl Stream/Edge first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream/Edge continues retrying the request without increasing the time between retries any further.

If the sender (which manages the connection to the Destination) is at capacity, it will not accept any incoming events. These incoming events originate internally from a previous stage of the data flow when Destinations send outbound requests to their respective external services, and they include retry requests and new requests. Any events that were already in transit when the sender reached capacity will continue to be processed downstream.

Sender capacity is freed up when an outgoing request succeeds or encounters a non-retryable error. When the sender has available capacity again, it will resume accepting incoming events. This capacity management is influenced by the number of active connections and configured limits, such as concurrency and buffer sizes. If a Pipeline sends events faster than the Destination can process, the buffers may fill up, leading to backpressure

and `Sender at capacity` warnings. This backpressure prevents the sender from accepting additional requests until capacity is restored.

For each response code you want to add to the list, select **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: Toggle on to automatically retry requests that have timed out and display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Toggle on to reject certificates that are **not** authorized by a CA in the **CA certificate path**, nor by another trusted CA (for example, the system's CA).

Round-robin DNS: Toggle on to enable round-robin DNS lookup across multiple IP addresses, IPv4 and IPv6. When a DNS server resolves a Fully Qualified Domain Name (FQDN) to multiple IP addresses, Cribl Edge will sequentially use each address in the order they are returned by the DNS server for subsequent connection attempts.

Compress: Defaults to toggled on, meaning compress the payload body before sending.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests per Worker Process. When Cribl Edge hits this limit, it begins throttling traffic to the downstream service. Defaults to `5`. Minimum: `1`. Maximum: `32`.

Body size limit (KB): Maximum size of the request body before compression. Defaults to `1024` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Body size limit** value until downstream receivers reliably accept all events.

Events-per-request limit: Maximum number of events to include in the request body. Defaults to `0`, allowing unlimited events.

Flush period (sec): Maximum time between requests. Low values can cause the payload size to be smaller than the configured **Body size limit**. Defaults to `1` second.

Extra HTTP headers: Click **Add Header** to insert extra headers as **Name/Value** pairs. Values will be sent encrypted.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Cribl Edge will redact all headers, except non-sensitive headers that you declare below in **Safe headers**.

Safe headers: Add headers to declare them as safe to log in plaintext. (Sensitive headers such as `authorization` will always be redacted, even if listed here.) Use a tab or hard return to separate header names.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Verifying the New Relic Events Destination

Once you've configured event sources, create one or more Routes to send data to New Relic.

In New Relic, you can create visualizations incorporating the Cribl Edge-supplied data, then add them to new or existing dashboards as widgets.

Alternatively, in the New Relic backend, you can select **Query you data** (top nav) > **Events** (left tab), and then select the event type you exported from Cribl Edge.

To view more events, change the time frame at the upper right. To see raw events, click **Raw data** on the right.

Proxying Requests

If you need to proxy HTTP/S requests, see System Proxy Configuration.

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

12.3.9.2. New Relic Logs & Metrics Destination

Cribl Edge supports sending events to the New Relic Log API and the New Relic Metric API.

Type: Streaming | TLS Support: Yes | PQ Support: Yes

In Cribl Edge 3.1.2 and newer, this Destination now authenticates using New Relic's Ingest License API key. (New Relic will retire the Insights Insert API keys, which this Destination previously used for authentication.)

Also in v.3.1.2 and newer, Cribl Edge provides a separate New Relic Events Destination that you can use to send ad hoc (loosely structured) events to New Relic via the New Relic Event API.

Within New Relic's platform, you can monitor Cribl Edge's performance and data flow by installing New Relic's Cribl dashboard.

Configure Cribl Edge to Output to New Relic

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**: **Output ID**: Enter a unique name to identify this New Relic definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
    - **Description**: Optionally, enter a description.
    - **Region**: Select which New Relic region endpoint to use. Select `Custom` to manually enter your New Relic region's URL if it's not listed.
3. Under **Authentication**, select an **Authentication method** from the dropdown:

If an incoming event contains an internal field named `__newRelic_apiKey`, the New Relic Logs & Metrics Destination uses that field's value as the API key when sending the event to New Relic.

> For events that do not contain an `__newRelic_apiKey` field, the Destination uses whatever API key you have configured in the **Authentication method** settings.

- **Manual**: This default option exposes an **API key** field. Directly enter your New Relic Ingest License API key, as you created or accessed it from New Relic's **account** drop-down. (For details, see the New Relic API Keys documentation.)
- **Secret**: This option exposes an **API key (text secret)** drop-down, in which you can select a stored secret that references a New Relic Ingest License API key. A **Create** link is available to store a new, reusable secret.

4. Next, you can configure the following **Optional Settings**:
   - **Log type**: Name of the `logType` to send with events. For example, `observability` or `access_log`.

     This sets a default. Where a `sourcetype` is specified in an event, it will override this value.
   - **Log message field**: Name of the field to send as the log `message` value. If not specified, the event will be serialized and sent as JSON.
   - **Fields**: Additional fields to (optionally) add, as **Name-Value** pairs. Click **Add Field** to add more. For details, see [Fields]{#fields}
   - **Backpressure behavior**: Select whether to block, drop, or queue events when all receivers are exerting backpressure. (Causes might include a broken or denied connection, or a rate limiter.) Defaults to `Block`. For the `Persistent Queue` option, see the section just below.
   - **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
5. Optionally, you can adjust the Persistent Queue, Processing, Retries, and Advanced settings outlined in the sections below.
6. Select **Save**, then **Commit & Deploy**.

Fields

These custom fields allow you to enrich your log data with specific details that are relevant to your use case. Click **Add Field** for more.

- **Name**: Enter the field name.
- **Value**:JavaScript expression to compute field's value, enclosed in single quotes, double quotes, or backticks. (Can evaluate to a constant.)

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.



If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5` GB. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.



Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Retries

Honor Retry-After header: Toggle on to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream/Edge limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. Any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. Toggle off to ignore all `Retry-After` headers.

Default Retry Behavior: By default, Cribl Stream/Edge applies predefined retry strategies for certain failed requests to the Destination using an exponential backoff algorithm. You can override these settings by manually configuring the retry behavior for a particular error code.

To retry requests, Cribl Stream/Edge applies the following rules by default:

Status Code	Action
Any in the `1xx, 3xx` series	Drop the request
`408, 429` errors	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached
Any other `4xx` series	Drop the request
Any `5xx` series	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached

By default, any `429 (Too Many Requests)` response that is not listed in the retry settings is treated as a non-retryable `4xx` error and dropped. If you want `429` responses retried, ensure `429` is included in the

Settings for failed HTTP requests.

Settings for failed HTTP requests: You can also manually configure Cribl Stream/Edge to automatically retry requests that receive particular HTTP response status codes.

Upon receiving a response code that's on the list, Cribl Stream/Edge first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream/Edge continues retrying the request without increasing the time between retries any further.

If the sender (which manages the connection to the Destination) is at capacity, it will not accept any incoming events. These incoming events originate internally from a previous stage of the data flow when Destinations send outbound requests to their respective external services, and they include retry requests and new requests. Any events that were already in transit when the sender reached capacity will continue to be processed downstream.

Sender capacity is freed up when an outgoing request succeeds or encounters a non-retryable error. When the sender has available capacity again, it will resume accepting incoming events. This capacity management is influenced by the number of active connections and configured limits, such as concurrency and buffer sizes. If a Pipeline sends events faster than the Destination can process, the buffers may fill up, leading to backpressure and `Sender at capacity` warnings. This backpressure prevents the sender from accepting additional requests until capacity is restored.

For each response code you want to add to the list, select **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: Toggle on to automatically retry requests that have timed out and display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.

- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Toggle on to reject certificates that are **not** authorized by a CA in the **CA certificate path**, nor by another trusted CA (for example, the system's CA).

Round-robin DNS: Toggle on to enable round-robin DNS lookup across multiple IP addresses, IPv4 and IPv6. When a DNS server resolves a Fully Qualified Domain Name (FQDN) to multiple IP addresses, Cribl Edge will sequentially use each address in the order they are returned by the DNS server for subsequent connection attempts.

Compress: Toggle on (default) to compress the payload body before sending.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests per Worker Process. When Cribl Edge hits this limit, it begins throttling traffic to the downstream service. Defaults to `5`. Minimum: `1`. Maximum: `32`.

Body size limit (KB): Maximum size of the request body before compression. Defaults to `1024` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Body size limit** value until downstream receivers reliably accept all events.

Buffer memory limit (KB): Total amount of memory used to buffer outgoing requests waiting to be sent. If left blank, defaults to 5 times the max body size (if set). If 0, no limit is enforced. This provides granular control over the memory allocated for buffering outgoing batched requests. Increasing the limit allows batches to grow larger before being flushed, improving efficiency for data with high cardinality (a large number of unique batches). Finding the optimal balance between efficient data transfer and memory usage involves adjusting both the **Buffer memory limit** and **Body size limit** settings.

Events-per-request limit: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values can cause the payload size to be smaller than the configured **Body size limit**. Defaults to `1` second.

Extra HTTP headers: Click **Add Header** to insert extra headers as **Name/Value** pairs. Values will be sent encrypted.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Cribl Edge will redact all headers, except non-sensitive headers that you declare below in **Safe headers**.

Safe headers: Add headers to declare them as safe to log in plaintext. (Sensitive headers such as `authorization` will always be redacted, even if listed here.) Use a tab or hard return to separate header names.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Verifying the New Relic Destination

Once you've configured log and/or metrics sources, create one or more Routes to send data to New Relic.

In New Relic, you can create visualizations incorporating the Cribl Edge-supplied data, then add them to new or existing dashboards as widgets.

Logs and metrics land in two different places in New Relic.

Log Queries

To access and query log data:

- Navigate to the New Relic home screen's **Logs** header option, and click the **(+)** button at right.
- Then to build your queries, use the **Find logs where** input field, and add desired columns to the table view below the graph,.

Metrics Queries

To access and query metrics data:

- From the New Relic home screen, *Click **Browse Data** > **Metrics** > **Can Search for metricNames**.
- Then, customize time range and dimensions to build the desired logic for your queries.
- Alternatively, you can use NRQL to build your own query searches.

Proxying Requests

If you need to proxy HTTP/S requests, see System Proxy Configuration.

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate,

12.3.10. Prometheus

You can send data from Cribl Edge to the following Prometheus services:

Destination	Description
Prometheus	Send metric events to Prometheus remote write targets
Grafana Cloud	Send data to Loki for logs and Prometheus for metrics
Loki	Send log events to Loki

12.3.10.1. Prometheus Destination

Cribl Edge can send metric events to targets and third-party platforms that support Prometheus' remote write specification (overview here).

Type: Streaming | TLS Support: Configurable | PQ Support: Yes

For an example of using this Destination to send host metrics to Grafana's Cribl Node Observability dashboard, see our System Metrics to Grafana topic.

Configure Cribl Edge to Output to Prometheus

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this Prometheus output definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
    - **Description**: Optionally, enter a description.
    - **Remote Write URL**: The endpoint to send events to, for example: `http://localhost:9200/write`
3. Next, you can configure the following **Optional Settings**:
    - **Backpressure behavior**: Whether to block, drop, or queue events when all receivers are exerting backpressure.
    - **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
4. Optionally, you can adjust the Authentication, Persistent Queue, Processing, Retries, and Advanced settings outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

Authentication

Select one of the following options for authentication:

- **None**: Don't use authentication.
- **Auth token**: Use HTTP token authentication. In the resulting **Token** field, enter the bearer token that must be included in the HTTP authorization header.
- **Auth token (text secret)**: This option exposes a **Token (text secret)** drop-down, in which you can select a stored text secret that references the bearer token described above. A **Create** link is available to store a new, reusable secret.
- **Basic**: Displays **Username** and **Password** fields for you to enter HTTP Basic authentication credentials.
- **Basic (credentials secret)**: This option exposes a **Credentials secret** drop-down, in which you can select a stored text secret that references the Basic authentication credentials described above. A **Create** link is available to store a new, reusable secret.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

> On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.
>
> This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.



If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5 GB`. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10

seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.



Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_host` (Cribl Edge Node that processed the event) and `cribl_wp` (Cribl Edge Worker Process that processed the event). Supports wildcards. Other options include:

- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_pipe` - Cribl Edge Pipeline that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.

Retries

Honor Retry-After header: Toggle on to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream/Edge limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. Any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. Toggle off to ignore all `Retry-After` headers.

Default Retry Behavior: By default, Cribl Stream/Edge applies predefined retry strategies for certain failed requests to the Destination using an exponential backoff algorithm. You can override these settings by manually configuring the retry behavior for a particular error code.

Destinations created before the 4.16.0 release do not support the default `4xx` series retry behavior. If your Destination was created prior to this release and you'd like it to support the behavior, reconfigure the Destination.

To retry requests, Cribl Stream/Edge applies the following rules by default:

Status Code	Action
Any in the `1xx, 3xx` series	Drop the request
`401, 403, 408, 429` errors	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached
Any other `4xx` series	Drop the request
Any in the `5xx` series	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached

By default, any `429 (Too Many Requests)` response that is not listed in the retry settings is treated as a non-retryable `4xx` error and dropped. If you want `429` responses retried, ensure `429` is included in the **Settings for failed HTTP requests**.

Settings for failed HTTP requests: You can also manually configure Cribl Stream/Edge to automatically retry requests that receive particular HTTP response status codes.

Upon receiving a response code that's on the list, Cribl Stream/Edge first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream/Edge continues retrying the request without increasing the time between retries any further.

If the sender (which manages the connection to the Destination) is at capacity, it will not accept any incoming events. These incoming events originate internally from a previous stage of the data flow when Destinations send outbound requests to their respective external services, and they include retry requests and new requests. Any events that were already in transit when the sender reached capacity will continue to be processed downstream.

Sender capacity is freed up when an outgoing request succeeds or encounters a non-retryable error. When the sender has available capacity again, it will resume accepting incoming events. This capacity management is influenced by the number of active connections and configured limits, such as concurrency and buffer sizes. If a Pipeline sends events faster than the Destination can process, the buffers may fill up, leading to backpressure and `Sender at capacity` warnings. This backpressure prevents the sender from accepting additional requests until capacity is restored.

For each response code you want to add to the list, select **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: Toggle on to automatically retry requests that have timed out and display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Reject certificates that are **not** authorized by a trusted CA (for example, the system's CA). Defaults to toggled on.

Round-robin DNS: Toggle on to enable round-robin DNS lookup across multiple IP addresses, IPv4 and IPv6. When a DNS server resolves a Fully Qualified Domain Name (FQDN) to multiple IP addresses, Cribl Edge will sequentially use each address in the order they are returned by the DNS server for subsequent connection attempts.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests per Worker Process. When Cribl Edge hits this limit, it begins throttling traffic to the downstream service. Defaults to `5`. Minimum: `1`. Maximum: `32`.

Body size limit (KB): Maximum size of the request body before compression. Defaults to `4096` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Body size limit** value until downstream receivers reliably accept all events.

Events-per-request limit: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Extra HTTP headers: Name-value pairs to pass as additional HTTP headers. Values will be sent encrypted.

Metric renaming expression: A JavaScript expression that can be used to rename metrics. The default expression - `name.replace(/\\./g, \'_\')` - replaces all `.` characters in a metric's name with the Prometheus-supported `_` character. Use the `name` global variable to access the metric's name. You can access event fields' values via `__e.<fieldName>`.

Send metadata: Whether to generate and send metrics' metadata (`type` and `metricFamilyName`) along with the metrics. Toggling on displays this additional field:

- **Metadata flush period (sec)**: How frequently metrics metadata is sent out. Value must at least equal the base **Flush period (sec)**. (In other words, metadata cannot be flushed on a shorter interval.) Defaults to `60` seconds.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Cribl Edge will redact all headers, except non-sensitive headers that you declare below in **Safe headers**.

Safe headers: Add headers to declare them as safe to log in plaintext. (Sensitive headers such as `authorization` will always be redacted, even if listed here.) Use a tab or hard return to separate header names.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Internal Fields

Cribl Edge uses a set of internal fields to assist in forwarding data to a Destination.

If an event contains the internal field `__criblMetrics`, Cribl Edge will send it to the HTTP endpoint as a metric event. Otherwise, Cribl Edge will drop the event.

Notes on HTTP-Based Outputs

- To proxy outbound HTTP/S requests, see System Proxy Configuration.
- Unlike other HTTP-based Destinations, Prometheus does not display an **Advanced Settings > Compress** option. The Prometheus `remote_write` spec requires payloads to be protobuf-encoded and snappy-

compressed, and data is always sent in this format. There is no separate user-configurable compression setting for this Destination as Cribl Stream/Edge automatically encodes and compresses every payload before sending.

- Cribl Edge will attempt to use keepalives to reuse a connection for multiple requests. After two minutes of the first use, the connection will be thrown away, and a new one will be reattempted. This is to prevent sticking to a particular destination when there is a constant flow of events.
- If the server does not support keepalives (or if the server closes a pooled connection while idle), a new connection will be established for the next request.
- When resolving the Destination's hostname, Cribl Edge will pick the first IP in the list for use in the next connection. Enable Round-robin DNS to better balance distribution of events between Prometheus cluster nodes.

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

12.3.10.2. Grafana Cloud Destination

Cribl Edge can send data to two of the services available in Grafana Cloud: Loki for logs and Prometheus for metrics. The Grafana Cloud Destination shapes events appropriately for Loki and Prometheus, and routes events to the correct endpoint for each service.



Type: Streaming | TLS Support: Configurable | PQ Support: Yes

For an alternative scenario that relies on our Prometheus Destination to send host metrics to Grafana's Cribl Node Observability dashboard, see our System Metrics to Grafana topic.

Preparing Prometheus and Loki to Receive Data from Cribl Edge

To define a Grafana Cloud Destination, you need a Grafana Cloud account.

While logged in to your Grafana account, navigate to the Grafana Cloud Portal, which should be located at `https://grafana.com/orgs/<your-organization-name>`, and complete the following steps.

Obtain an API key, setting its Role to `MetricsPublisher`. If you want Cribl Edge or an external KMS to manage the API key, configure a key pair that references the API key.

In the Prometheus tile, select **Send Metrics** to open the Prometheus configuration page. Write down:

- Your **Remote Write Endpoint** URL, for example: `https://prometheus-blocks-prod-us-central1.grafana.net/api/prom/push`.
- Your Prometheus **Username**.

In the Loki tile, select **Send Logs** to open the Loki configuration page. Write down:

- Your **Grafana Data Source settings** URL, for example: `https://logs-prod-us-central1.grafana.net`.
- Your Loki **User** ID.

Decide what type of authentication to use and prepare accordingly:

- If you choose Basic authentication, the username (**Username** in Prometheus, **User** in Loki) and password (simply your Grafana API key) will remain separate.
- If you choose token-based authentication, construct your tokens by concatenating username, colon (`:`), and password, for example `12345:cOQvDj6sJGFS3Bk2MguBW==`. Because the Prometheus and Loki

usernames differ, you need to construct a separate token for each service.

Configure Cribl Edge to Output to Grafana Cloud

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this Grafana Cloud output definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
    - **Description**: Optionally, enter a description.
    - **Loki URL**: The endpoint to send log events to, for example: `https://logs-prod-us-central1.grafana.net`. This is the **Grafana Data Source settings** URL you wrote down earlier.
    - **Prometheus URL**: The endpoint to send metric events to, for example: `https://prometheus-blocks-prod-us-central1.grafana.net/api/prom/push`. This is the **Remote Write Endpoint** URL you wrote down earlier.
3. Next, you can configure the following **Optional Settings**:
    - **Backpressure behavior**: Whether to block, drop, or queue events when all receivers are exerting backpressure.
    - **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
4. Optionally, you can adjust the Persistent Queue, Authentication, Processing, Retries, and Advanced settings outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5 GB`. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.



Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Authentication

The **Authentication** tab provides separate **Loki** and **Prometheus** sections, enabling you to configure these inputs separately. The two sections provide identical options.

Select one of the following options for authentication:

- **Auth token**: Enter the bearer token that must be included in the authorization header. Use the token that you constructed earlier. In Grafana Cloud, the bearer token is generally built by concatenating the username and the API key, separated by a colon. For example: `<your-username>:<your-api-key>`.
- **Auth token (text secret)**: This option exposes a drop-down in which you can select a stored text secret that references the bearer token described above. A **Create** link is available to store a new, reusable secret.
- **Basic**: This default option displays fields for you to enter HTTP Basic authentication credentials. **Username** is the Loki **User** or Prometheus **Username** that you wrote down earlier. **Password** is your API key in the Grafana Cloud domain.
- **Basic (credentials secret)**: This option exposes a **Credentials secret** drop-down, in which you can select a stored text secret that references the Basic authentication credentials described above. A **Create** link is available to store a new, reusable secret.

Processing Settings

Metric events can have dimensions, and log events have labels. Dimensions, labels, and their values are determined by several different settings in Cribl Edge. This section explains how that works, along with other kinds of settings.

Loki uses labels to define separate streams of logging data. This is a key concept. Cribl recommends that you familiarize yourself with the information and documentation Grafana provides about labels in Loki.

One canonical example is processing logs from servers in three environments: production, staging, and testing. You could create a label named `env` whose possible values are `prod`, `staging`, and `test`.

One basic principle is that if you set too many labels, you can end up with too many streams.

Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output–both metric events, as dimensions; and, log events, as labels. Supports wildcards.

By default, includes `cribl_host` (Cribl Edge Node that processed the event) and `cribl_wp` (Cribl Edge Worker Process that processed the event). On the Loki side, this creates different streams, which prevents Loki from rejecting some events as being out of order when different Nodes or Worker Processes are emitting at different rates.

Other options include:

- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.

- `cribl_pipe` - Cribl Edge Pipeline that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.

Retries

Honor Retry-After header: Toggle on to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream/Edge limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. Any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. Toggle off to ignore all `Retry-After` headers.

Default Retry Behavior: By default, Cribl Stream/Edge applies predefined retry strategies for certain failed requests to the Destination using an exponential backoff algorithm. You can override these settings by manually configuring the retry behavior for a particular error code.

To retry requests, Cribl Stream/Edge applies the following rules by default:

Status Code	Action
Any in the `1xx, 3xx` series	Drop the request
`408, 429` errors	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached
Any other `4xx` series	Drop the request
Any `5xx` series	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached

By default, any `429 (Too Many Requests)` response that is not listed in the retry settings is treated as a non-retryable `4xx` error and dropped. If you want `429` responses retried, ensure `429` is included in the **Settings for failed HTTP requests**.

Settings for failed HTTP requests: You can also manually configure Cribl Stream/Edge to automatically retry requests that receive particular HTTP response status codes.

Upon receiving a response code that's on the list, Cribl Stream/Edge first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream/Edge continues retrying the request without increasing the time between retries any further.

If the sender (which manages the connection to the Destination) is at capacity, it will not accept any incoming events. These incoming events originate internally from a previous stage of the data flow when Destinations send outbound requests to their respective external services, and they include retry requests and new

requests. Any events that were already in transit when the sender reached capacity will continue to be processed downstream.

Sender capacity is freed up when an outgoing request succeeds or encounters a non-retryable error. When the sender has available capacity again, it will resume accepting incoming events. This capacity management is influenced by the number of active connections and configured limits, such as concurrency and buffer sizes. If a Pipeline sends events faster than the Destination can process, the buffers may fill up, leading to backpressure and `Sender at capacity` warnings. This backpressure prevents the sender from accepting additional requests until capacity is restored.

For each response code you want to add to the list, select **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: Toggle on to automatically retry requests that have timed out and display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (for example, the system's CA). Default is toggled on.

Round-robin DNS: Toggle on to enable round-robin DNS lookup across multiple IP addresses, IPv4 and IPv6. When a DNS server resolves a Fully Qualified Domain Name (FQDN) to multiple IP addresses, Cribl Edge will sequentially use each address in the order they are returned by the DNS server for subsequent connection attempts.

Send structured metadata: Adds key-value pairs from the `__structuredMetadata` field to logs as structured metadata. The field can be from a Source or added in a Pipeline. Non-string values are automatically converted to strings. To avoid unexpected results from this conversion, we recommend providing values as strings.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests before blocking. This is set per Worker Process. Defaults to `5`.

Body size limit (KB): Maximum size of the request body before compression. Defaults to `4096` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Body size limit** value until downstream receivers reliably accept all events.

Events-per-request limit: Maximum number of events to include in the request body. The `0` default allows unlimited events.



Loki and Prometheus might complain about entries being delivered out of order when **Request concurrency** is set > `1` and any of **Flush period (sec)**, **Body size limit (KB)**, or **Max events per request** are set to low values.

Flush period (sec): Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Extra HTTP headers: Name-value pairs to pass as additional HTTP headers. Values will be sent encrypted.

Metric renaming expression: A JavaScript expression that can be used to rename metrics. The default expression - `name.replace(/\\./g, \'_\')` - replaces all `.` characters in a metric's name with the Prometheus-supported `_` character. Use the `name` global variable to access the metric's name. You can access event fields' values via `__e.<fieldName>`.

Message format: Whether to send events as `Protobuf` (the default) or `JSON`.

Compress: When the **Message format** is `JSON`, this setting controls the data compression format used before sending the data to Grafana Cloud. Defaults to toggled on for GZIP-compression. (Applies only to Loki's JSON payloads. This toggle is hidden when the **Message format** is `Protobuf`, because both Prometheus' and Loki's Protobuf implementations are Snappy-compressed by default.)

Logs message field: The event field to send as log output, for example: `_raw`. All other event fields are discarded. If left blank, Cribl Edge sends a JSON representation of the whole event.

Logs labels: Name/value pairs where the value can be a static or dynamic expression that has access to all log event fields.

Failed request logging mode: Determines which data is logged when a request fails. Use the drop-down to select one of these options:

- `None` (default).
- `Payload`.
- `Payload + Headers`. Use the **Safe Headers** field below to specify the headers to log. If you leave that field empty, all headers are redacted, even with this setting.

Safe headers: List the headers you want to log, in plain text.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Internal Fields

Cribl Edge uses a set of internal fields to assist in forwarding data to a Destination.

If an event contains the internal field `__criblMetrics`, Cribl Edge will send it Prometheus as a metric event. If `__criblMetrics` is absent, Cribl Edge will treat the event as a log and send it to Loki.

The internal field `__structuredMetadata` is referenced when the **Send structured metadata** advanced setting is toggled on.

Labels:

The internal field `__labels` specifies labels to add to log events. If a label is set in both the `__labels` field and in **Logs labels** and/or **System fields**, Cribl Edge sends the value from `__labels` to Loki. Setting the `__labels` field in a Pipeline gives you a quick way to experiment with the logs being sent.

If there are no labels set (this would happen when **System fields**, **Logs labels**, and `__labels` are all empty), Cribl Edge adds a default `source` label, which prevents Loki from rejecting events. The `source` label the concatenation of `cribl`, underscore (`_`), source type, colon (`:`), source-name, where source name and type are values in the `__inputId` event field, for example: `cribl_metrics:in_prometheus_rw`. If `__inputId` is missing, `source` is set to `cribl`.

Notes on HTTP-Based Outputs

- To proxy outbound HTTP/S requests, see System Proxy Configuration.
- The **Advanced Settings > Compress** toggle determines whether to compress the payload body before sending to Loki only. The toggle setting does not apply to Prometheus payloads, which are always compressed using Snappy.

- Cribl Edge will attempt to use keepalives to reuse a connection for multiple requests. After two minutes of the first use, the connection will be thrown away, and a new one will be reattempted. This is to prevent sticking to a particular destination when there is a constant flow of events.
- If the server does not support keepalives (or if the server closes a pooled connection while idle), a new connection will be established for the next request.
- When resolving the Destination's hostname, Cribl Edge will pick the first IP in the list for use in the next connection. Enable Round-robin DNS to better balance distribution of events between Grafana Cloud nodes.

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

12.3.10.3. Loki Destination

Cribl Edge can send log events to Grafana's Loki log aggregation system.



Type: Streaming | TLS Support: Yes | PQ Support: Yes

Configuring Cribl Edge to Output to Loki

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this Loki output definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
    - **Description**: Optionally, enter a description.
    - **Loki URL**: The endpoint to send events to, for example: `https://logs-prod-us-central1.grafana.net`.

        To proxy outbound HTTP/S requests, see System Proxy Configuration.
3. Next, you can configure the following **Optional Settings**:
    - **Backpressure behavior**: Whether to block, drop, or queue events when all receivers are exerting backpressure.
    - **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
4. Optionally, you can adjust the Authentication, Persistent Queue, Processing, Retries, and Advanced settings outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

Authentication

Select one of the following options for authentication:

- **None**: Don't use authentication.
- **Auth token**: Use HTTP token authentication. In the resulting **Token** field, enter the bearer token that must be included in the HTTP authorization header.
- **Auth token (text secret)**: This option exposes a drop-down in which you can select a stored text secret that references the bearer token described above. A **Create** link is available to store a new, reusable secret.
- **Basic**: This default option displays fields for you to enter HTTP Basic authentication credentials. **Username** is the Loki **User**. **Password** is your API key in the Grafana Cloud domain.
- **Basic (credentials secret)**: This option exposes a **Credentials secret** drop-down, in which you can select a stored text secret that references the Basic authentication credentials described above. A **Create** link is available to store a new, reusable secret.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

> On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.
>
> This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.



If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5 GB`. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10

seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.



Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Processing Settings

Loki uses labels to define separate streams of logging data. This is a key concept. Cribl recommends that you familiarize yourself with the information and documentation Grafana provides about labels in Loki.

One canonical example is processing logs from servers in three environments: production, staging, and testing. You could create a label named `env` whose possible values are `prod`, `staging`, and `test`.

One basic principle is that if you set too many labels, you can end up with too many streams.

Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to log events as labels. Supports wildcards.

By default, includes `cribl_host` (Cribl Edge Node that processed the event) and `cribl_wp` (Cribl Edge Worker Process that processed the event). On the Loki side, this creates different streams, which prevents Loki from rejecting some events as being out of order when different Nodes or Worker Processes are emitting at different rates.

Other options include:

- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_pipe` - Cribl Edge Pipeline that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.

Retries

Honor Retry-After header: Toggle on to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream/Edge limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. Any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. Toggle off to ignore all `Retry-After` headers.

Default Retry Behavior: By default, Cribl Stream/Edge applies predefined retry strategies for certain failed requests to the Destination using an exponential backoff algorithm. You can override these settings by manually configuring the retry behavior for a particular error code.

To retry requests, Cribl Stream/Edge applies the following rules by default:

Status Code	Action
Any in the `1xx`, `3xx` series	Drop the request
`408`, `429` errors	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached
Any other `4xx` series	Drop the request
Any `5xx` series	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached

By default, any `429 (Too Many Requests)` response that is not listed in the retry settings is treated as a non-retryable `4xx` error and dropped. If you want `429` responses retried, ensure `429` is included in the **Settings for failed HTTP requests**.

Settings for failed HTTP requests: You can also manually configure Cribl Stream/Edge to automatically retry requests that receive particular HTTP response status codes.

Upon receiving a response code that's on the list, Cribl Stream/Edge first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream/Edge continues retrying the request without increasing the time between retries any further.

If the sender (which manages the connection to the Destination) is at capacity, it will not accept any incoming events. These incoming events originate internally from a previous stage of the data flow when Destinations send outbound requests to their respective external services, and they include retry requests and new requests. Any events that were already in transit when the sender reached capacity will continue to be processed downstream.

Sender capacity is freed up when an outgoing request succeeds or encounters a non-retryable error. When the sender has available capacity again, it will resume accepting incoming events. This capacity management is influenced by the number of active connections and configured limits, such as concurrency and buffer sizes. If a Pipeline sends events faster than the Destination can process, the buffers may fill up, leading to backpressure and `Sender at capacity` warnings. This backpressure prevents the sender from accepting additional requests until capacity is restored.

For each response code you want to add to the list, select **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: Toggle on to automatically retry requests that have timed out and display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Round-robin DNS: Toggle on to enable round-robin DNS lookup across multiple IP addresses, IPv4 and IPv6. When a DNS server resolves a Fully Qualified Domain Name (FQDN) to multiple IP addresses, Cribl Edge will sequentially use each address in the order they are returned by the DNS server for subsequent connection attempts.

Validate server certs: Reject certificates that are not authorized by a trusted CA (for example, the system's CA). Defaults to toggled on.

Send structured metadata: Adds key-value pairs from the `__structuredMetadata` field to logs as structured metadata. The field can be from a Source or added in a Pipeline. Non-string values are automatically converted

to strings. To avoid unexpected results from this conversion, we recommend providing values as strings.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests before blocking. This is set per Worker Process. Defaults to `1`.

Body size limit (KB): Maximum size of the request body before compression. Defaults to `4096` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Body size limit** value until downstream receivers reliably accept all events.

Buffer memory limit (KB): Total amount of memory used to buffer outgoing requests waiting to be sent. If left blank, defaults to 5 times the max body size (if set). If 0, no limit is enforced. This provides granular control over the memory allocated for buffering outgoing batched requests. Increasing the limit allows batches to grow larger before being flushed, improving efficiency for data with high cardinality (a large number of unique batches). Finding the optimal balance between efficient data transfer and memory usage involves adjusting both the **Buffer memory limit** and **Body size limit** settings.

Events-per-request limit: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `15`.

Extra HTTP headers: Name-value pairs to pass as additional HTTP headers. Values will be sent encrypted.

Enable dynamic headers: Toggle on to enable dynamic HTTP headers where the Destination checks each event for the internal field `__headers`. Events with the same `__headers` are batched and sent together in a single HTTP request. Events with different `__headers` are sent in separate requests. Use this when you need per-event custom headers, such as for multi-tenant routing or dynamic authentication.

Message format: Whether to send events as `Protobuf` (the default) or `JSON`.

Compress: When the **Message format** is `JSON`, this setting controls the data compression format used before sending the data to Loki. Defaults to toggled on for GZIP-compression. (Applies only to Loki's JSON payloads. This toggle is hidden when the **Message format** is `Protobuf`, because both Prometheus' and Loki's Protobuf implementations are Snappy-compressed by default.)

Logs message field: The event field to send as log output, for example: `_raw`. All other event fields are discarded. If left blank, Cribl Edge sends a JSON or Protobuf representation of the whole event.

Logs labels: Name/value pairs where the value can be a static or dynamic expression that has access to all log event fields.

Failed request logging mode: Determines which data is logged when a request fails. Use the drop-down to select one of these options:

- `None` (default).
- `Payload`.
- `Payload + Headers`. Use the **Safe Headers** field below to specify the headers to log. If you leave that field empty, all headers are redacted, even with this setting.

Safe headers: List the headers you want to log, in plain text.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Internal Fields

Cribl Edge uses internal fields to assist in forwarding data to a Destination.

- `__labels` specifies labels to add to log events. If a label is set in both the `__labels` field and in Logs labels and/or System fields, Cribl Stream sends the value from `__labels` to Loki. Setting the `__labels` field in a Pipeline gives you a quick way to experiment with the logs being sent.
- `__structuredMetadata` is referenced when the **Send structured metadata** advanced setting is toggled on.
- `__headers` allows you to define custom HTTP headers on a per-event basis when the **Enable dynamic headers** advanced setting is toggled on. Top-level string key-value pairs are supported. Non-string values are ignored.

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

12.3.11. Splunk

You can send data from Cribl Edge to the following Splunk services:

Destination	Description
Splunk HEC	Stream data to a Splunk HEC receiver
Splunk Single Instance	Stream data to a Splunk instance
Splunk Load Balanced	Load-balance data streaming to multiple Splunk receivers

12.3.11.1. Splunk HEC Destination

The **Splunk HEC** Destination can stream data to a Splunk HEC (HTTP Event Collector) receiver through the event, raw, and S2S endpoints.

The data arrives to Splunk cooked and parsed, so it enters at the Splunk data pipeline's indexing segment.



Type: Streaming | TLS Support: Yes | PQ Support: Yes

Looking for a quick way to switch all of your S2S Destinations to Splunk HEC Destinations? Follow our how-to guide: Switch Cribl Stream Destinations from S2S to Splunk HEC.

The Splunk HEC Destination is recommended for Splunk Cloud. For details, see Splunk Cloud Platform and BYOL Integrations.

Configure Cribl Edge to Output to Splunk HEC Destinations

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this Splunk HEC definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
    - **Description**: Optionally, enter a description.
    - **Load balancing**: Toggle on to specify multiple Splunk HEC endpoints and load weights. If toggled off and you notice that Cribl Edge is not sending data to all possible IP addresses, enable **Advanced Settings** > **Round-robin DNS**.
    - **Splunk HEC endpoint(s)**: URLs of Splunk HEC endpoints to send events to.
3. Under **Authentication**, select an **Authentication method** from the dropdown:
    - **Manual**: Displays an **HEC Auth token** field for you to enter your Splunk HEC token.

- **Secret**: This option exposes an **HEC Auth token (text secret)** drop-down, in which you can select a stored secret that references the HEC token described above. A **Create** link is available to store a new, reusable secret.

4. Next, you can configure the following **Optional Settings**:
   - **Exclude current host IPs**: This toggle appears when **Load balancing** is toggled on. It determines whether to exclude all IPs of the current host from the list of any resolved hostnames. Default is toggled off, which keeps the current host available for load balancing.
   - **Backpressure behavior**: Select whether to block, drop, or queue events when all receivers are exerting backpressure. (Causes might include a broken or denied connection, or a rate limiter.) Defaults to `Block`.
   - **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
5. Optionally, you can adjust the Persistent Queue, TLS, Processing, Retries and Advanced settings outlined in the sections below.



If you're sending data to Splunk Cloud, Cribl recommends a maximum value of `1 MB` for **Body size limit (KB)** under Advanced settings. This upper limit is defined by the maximum content length for the default HTTP Event Collector (HEC) provided by Splunk Cloud.

6. Select **Save**, then **Commit & Deploy**.

Splunk HEC Endpoints

Use the **Splunk HEC Endpoints** table to specify a known set of receivers on which to load-balance data. The table appears only when **Load balancing** is toggled on.

To specify more receivers on new rows, select **Add Endpoint**. Each row provides the following fields:

- **HEC Endpoint**: Specify the URL to a Splunk HEC endpoint to send events to. See the Splunk documentation for details on getting your URL. For Splunk Cloud endpoints, `https` protocol is recommended. Supported endpoints include:
  - `/services/collector/event` (default)
  - `/services/collector/raw`
  - `/services/collector/s2s`
- **Load Weight**: Set the relative traffic-handling capability for each connection by assigning a weight (`> 0`). This column accepts arbitrary values, but for best results, assign weights in the same order of magnitude to all connections. Cribl Edge will attempt to distribute traffic to the connections according to their relative weights.

The final column provides an `X` button to delete any row from the table.

When you first enable load balancing, or if you edit the load weight once your data is load-balanced, give the logic time to settle. The changes might take a few seconds to register.

For details on configuring all these options, see About Load Balancing.

Indexer Acknowledgement

Do not toggle **Enable Indexer Acknowledgement** on for the Splunk token. With this toggle on, Splunk receivers expect the Channel GUID to be passed in, and requests will fail with errors of this form:

```
channel: output:splunk_cloud_hec
cid: w2
level: error
message: Request failed
reason: Received status code=400, method=POST, url=https://http-inputs-bazookatron.sp
response: {"text":"Data channel is missing","code":10}
time: 2023-02-14T15:26:03.413Z
```

Copy

TLS Settings (Client Side)

Enabled: Defaults to toggled off. When toggled on:

Server name (SNI): Server name for the SNI (Server Name Indication) TLS extension. This must be a hostname, not an IP address.

Minimum TLS version: Optionally, select the minimum TLS version to use when connecting.

Maximum TLS version: Optionally, select the maximum TLS version to use when connecting.

Certificate: Select an existing certificate name from the drop-down or add a new certificate by selecting **Create**. See Add a TLS Certificate to a Fleet for details.

CA certificate path: Path on client containing CA certificates (in PEM format) to use to verify the server's cert. Path can reference `$ENV_VARS`.

By default, this Destination rejects certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (for example, the system's CA). You can control this behavior using the

Validate server certs advanced setting.

Private key path (mutual auth): Path on client containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Certificate path (mutual auth): Path on client containing certificates in (PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Passphrase: Passphrase to use to decrypt private key. Often the value of the `sslPassword` or similar parameter in the `outputs.conf` or `server.conf` file.

Single .pem File

If you have a **single** .pem file containing `cacert`, `key`, and `cert` sections, enter it in all of these fields above: **CA certificate path**, **Private key path (mutual auth)**, and **Certificate path (mutual auth)**.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.



If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5 GB`. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10

seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.



Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Retries

Honor Retry-After header: Toggle on to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream/Edge limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. Any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. Toggle off to ignore all `Retry-After` headers.

Default Retry Behavior: By default, Cribl Stream/Edge applies predefined retry strategies for certain failed requests to the Destination using an exponential backoff algorithm. You can override these settings by manually configuring the retry behavior for a particular error code.

Destinations created before the 4.16.0 release do not support the default `4xx` series retry behavior. If your Destination was created prior to this release and you'd like it to support the behavior, reconfigure the Destination.

To retry requests, Cribl Stream/Edge applies the following rules by default:

Status Code	Action
Any in the `1xx, 3xx` series	Drop the request
`401, 403, 408, 429` errors	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached
Any other `4xx` series	Drop the request
Any in the `5xx` series	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached

By default, any `429 (Too Many Requests)` response that is not listed in the retry settings is treated as a non-retryable `4xx` error and dropped. If you want `429` responses retried, ensure `429` is included in the **Settings for failed HTTP requests**.

Settings for failed HTTP requests: You can also manually configure Cribl Stream/Edge to automatically retry requests that receive particular HTTP response status codes.

Upon receiving a response code that's on the list, Cribl Stream/Edge first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream/Edge continues retrying the request without increasing the time between retries any further.

If the sender (which manages the connection to the Destination) is at capacity, it will not accept any incoming events. These incoming events originate internally from a previous stage of the data flow when Destinations send outbound requests to their respective external services, and they include retry requests and new requests. Any events that were already in transit when the sender reached capacity will continue to be processed downstream.

Sender capacity is freed up when an outgoing request succeeds or encounters a non-retryable error. When the sender has available capacity again, it will resume accepting incoming events. This capacity management is influenced by the number of active connections and configured limits, such as concurrency and buffer sizes. If a Pipeline sends events faster than the Destination can process, the buffers may fill up, leading to backpressure and `Sender at capacity` warnings. This backpressure prevents the sender from accepting additional requests until capacity is restored.

For each response code you want to add to the list, select **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: Toggle on to automatically retry requests that have timed out and display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Output multi-metrics: Toggle on to output multiple-measurement metric data points. (Supported in Splunk 8.0 and above, this format enables sending multiple metrics in a single event, improving the efficiency of your Splunk capacity.)

Validate server certs: Reject certificates that are not authorized by a trusted CA (for example, the system's CA). Defaults to toggled on.

Round-robin DNS: Toggle on to enable round-robin DNS lookup across multiple IP addresses, IPv4 and IPv6. When a DNS server resolves a Fully Qualified Domain Name (FQDN) to multiple IP addresses, Cribl Edge will sequentially use each address in the order they are returned by the DNS server for subsequent connection attempts. (This setting is available only when **General Settings** > **Load balancing** is toggled off.)

Compress: Toggle on (default) to compress the payload body before sending.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests per Worker Process. When Cribl Edge hits this limit, it begins throttling traffic to the downstream service. Defaults to `5`. Minimum: `1`. Maximum: `32`. Each request can potentially hit a different HEC receiver.

Body size limit (KB): Maximum size, in KB, of the request body. Defaults to `4096`. Lowering the size can potentially result in more parallel requests and also cause outbound requests to be made sooner.

If you're sending data to Splunk Cloud, Cribl recommends a maximum value of `1 MB` for **Body size limit (KB)**. This upper limit is defined by the maximum content length for the default HTTP Event Collector (HEC) provided by Splunk Cloud.

Events-per-request limit: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values can cause the payload size to be smaller than the configured **Body size limit**. Defaults to `1`.

- Retries occur on the configured flush interval.
- Any HTTP response code in the `2xx` range is considered success.
- Any response code in the `5xx` range is considered a retryable error, which will not trigger Persistent Queue (PQ) usage.
- `4xx` responses are dropped by default and do not trigger PQ. To retry specific `4xx` codes (for example, `429`), add them explicitly under **Retries**. When configured as retryable, those responses will engage PQ (if PQ is configured as the Backpressure behavior).

Extra HTTP headers: Select **Add Header** to add **Name**/**Value** pairs to pass as additional HTTP headers. Values will be sent encrypted.

The next two fields appear only when you toggle **Load balancing** on in the **General Settings**.

DNS resolution period (seconds): Re-resolve any hostnames after each interval of this many seconds, and pick up destinations from A records. Defaults to `600` seconds.

Load balance stats period (seconds): The duration (in seconds) Cribl Edge tracks historical traffic data to influence load balancing decisions. Defaults to `300` seconds. If a hostname resolves to multiple IPs (via DNS A records), all IPs inherit the weight assigned to the hostname unless each IP is explicitly configured with its own weight. Cribl Edge prioritizes IP-specific settings over hostname-derived settings for load balancing.

Throttling: The rate (in bytes per second) to throttle while writing to the Destination. Enter `0` (the default) for no limit, or a value in multi-byte units such as KB, MB, or GB. When outbound traffic reaches this limit, Cribl Edge slows down sending to this Destination. Any additional data follows the existing **Backpressure behavior** and **Persistent Queue** settings.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Cribl Edge will redact all headers, except non-sensitive headers that you declare below in **Safe headers**.

Safe headers: Add headers to declare them as safe to log in plaintext. (Sensitive headers such as `authorization` will always be redacted, even if listed here.) Use a tab or hard return to separate header names.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Splunk App Settings provides two fields that take effect only in the Cribl App for Splunk. (Cribl Edge ignores their values.)

- **Next processing queue**: Specify the next Splunk processing queue to send the events to, after HEC processing. Defaults to `indexQueue`.
- **Default _TCP_ROUTING**: Specify the value of the `_TCP_ROUTING` field for events that do not have `_ctrl._TCP_ROUTING` set. Defaults to `nowhere`. This is useful only when you expect the HEC receiver to route this data on to another destination.

Event Format

Cribl Edge uses the internal field `__criblMetrics` to determine whether to process an event as a metric or a log. See Splunk's documentation on their event format for HEC events.

Metric event: When an event contains the `__criblMetrics` field, the Destination constructs a JSON object following Splunk's metric event schema.

- Any key-value pairs in `__criblMetrics.dims` are added as dimension fields to the final Splunk metric event.
- Metrics listed in `__criblMetrics.values` are formatted based on the **Output multi-metrics** advanced setting:
    - **Enabled**: A single event is created that bundles multiple metrics using the `metric_name:<name>` format for each value.
    - **Disabled**: A separate event is created for each metric, using the standard `metric_name` and `_value` fields.

Log event: If an event does not contain the `__criblMetrics` field, it's processed as a log event.

- If `_raw` is present, it becomes the value of the `event` property in the JSON payload.
- If `_raw` is absent, the entire event (minus internal fields) is serialized as a JSON object and used as the `event` property.

Notes on HTTP-Based Outputs

- To proxy outbound HTTP/S requests, see System Proxy Configuration.
- Cribl Edge will attempt to use keepalives to reuse a connection for multiple requests. After two minutes of the first use, the connection will be thrown away, and a new connection will be reattempted. This is to prevent sticking to a particular Destination when there is a constant flow of events.
- If the server does not support keepalives - or if the server closes a pooled connection while idle - a new connection will be established for the next request.
- When resolving the Destination's hostname with load balancing disabled, Cribl Stream will pick the first IP in the list for use in the next connection. Enable Round-robin DNS to better balance distribution of events between Splunk HEC servers.
- See Splunk's documentation on editing `fields.conf` to ensure the visibility of index-time fields sent to Splunk by Cribl Edge.

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

Cribl University offers an Advanced Troubleshooting > Destination Integrations: Splunk Cloud short course. To follow the direct course link, first log into your Cribl University account. (To create an account, select the **Sign up** link. You'll need to select through a short **Terms & Conditions** presentation, with chill music, before proceeding to courses - but Cribl's training is always free of charge.) Once logged in, check out other useful Advanced Troubleshooting short courses and Troubleshooting Criblets.

12.3.11.2. Splunk Single Instance Destination

This Splunk Destination can stream data to a **free** Splunk Cloud instance. From the perspective of the receiving Splunk Cloud instance, the data arrives cooked and parsed.



Type: Streaming | TLS Support: Configurable | PQ Support: Yes

Looking for a quick way to switch all of your S2S Destinations to Splunk HEC Destinations? Follow our how-to guide: Switch Cribl Stream Destinations from S2S to Splunk HEC.

Splunk Cloud Platform

To establish secure connections to Splunk Cloud, you'll need to configure SSL settings using the private and public keys from the Splunk Cloud Universal Forwarder credentials package.

For details on which Destination to use for Splunk Cloud, see Splunk Cloud Platform and BYOL Integrations. The Splunk HEC Destination is recommended for most use cases.

Configure Cribl Edge to Output to Splunk Destinations

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this Splunk Single Instance definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
    - **Description**: Optionally, enter a description.
    - **Address**: Hostname of the Splunk receiver.
    - **Port**: The port number on the host.
3. Next, you can configure the following **Optional Settings**:

- **Backpressure behavior**: Select whether to block, drop, or queue events when all receivers are exerting backpressure. (Causes might include a broken or denied connection, or a rate limiter.) Defaults to `Block`.
- **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

4. Optionally, you can adjust the Persistent Queue, TLS, Processing, Timeout, and Advanced settings outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

> On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.
>
> This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.

Mode	Description
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.



If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5` GB. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.



Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

TLS Settings (Client Side)

Enabled: Defaults to toggled off. When toggled on:

Validate server certs: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (for example, the system's CA). Defaults to toggled on.

Server name (SNI): Server name for the SNI (Server Name Indication) TLS extension. This must be a hostname, not an IP address.

Minimum TLS version: Optionally, select the minimum TLS version to use when connecting.

Maximum TLS version: Optionally, select the maximum TLS version to use when connecting.

Certificate: Select an existing certificate name from the drop-down or add a new certificate by selecting **Create**. See Add a TLS Certificate to a Fleet for details.

CA certificate path: Path on client containing CA certificates (in PEM format) to use to verify the server's cert. Path can reference `$ENV_VARS`.

Private key path (mutual auth): Path on client containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Certificate path (mutual auth): Path on client containing certificates in (PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Passphrase: Passphrase to use to decrypt private key. Often the value of the `sslPassword` or similar parameter in the `outputs.conf` or `server.conf` file.



Single .pem File

> If you have a **single** .pem file containing `cacert`, `key`, and `cert` sections, enter it in all of these fields above: **CA certificate path**, **Private key path (mutual auth)**, and **Certificate path (mutual auth)**.

Timeout Settings

Connection timeout: Amount of time (in milliseconds) to wait for the connection to establish, before retrying. Defaults to `10000`.

Write timeout: Amount of time (in milliseconds) to wait for a write to complete, before assuming connection is dead. Defaults to `60000`.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Advanced Settings

Output multiple metrics: Toggle on to output multiple-measurement metric data points. (Supported in Splunk 8.0 and above, this format enables sending multiple metrics in a single event, improving the efficiency of your Splunk capacity.)

Minimize in-flight data loss: If toggled on (default), Cribl Edge will check whether the indexer is shutting down, and if so, will stop sending data. This helps minimize data loss during shutdown. (Note that Splunk logs will indicate that the Cribl app has set `UseAck` to `true`, even though Cribl does not enable full `UseAck` behavior.) If toggled off, exposes the following alternative option:

Failed health check limit: Displayed (and set to `1` by default) only if **Minimize in-flight data loss** is disabled. This option sends periodic requests to Splunk once per minute, to verify that the Splunk endpoint is still alive and can receive data. Its value governs how many failed requests Cribl Edge will allow before closing this connection.

A low threshold value improves connections' resilience, but by proliferating connections, this can complicate troubleshooting. Set to `0` to disable health checks entirely - here, if the connection to Splunk is forcibly closed, you risk some data loss.

Max S2S version: The highest version of the Splunk-to-Splunk protocol to expose during handshake. Defaults to `v4`; `v3` is also available.

Throttling: Throttle rate, in bytes per second. Defaults to `0`, meaning no throttling. Multiple-byte units such as KB, MB, GB, and so forth, are also allowed - for example, `42 MB`. When throttling is engaged, your **Backpressure behavior** selection determines whether Cribl Edge will handle excess data by blocking it, dropping it, or queueing it to disk.

- When used in conjunction with **Connection limit**, the throttling rate is applied per connection. For instance, if **Connection limit** is set to `2` and throttling is set to `3 MB`, each of the two connections can send data at a rate of up to 3 MB per second, resulting in a total potential throughput of 6 MB per second across both connections.

Nested field serialization: Specifies how to serialize nested fields into index-time fields. Defaults to `None`.

Log failed requests to disk: Toggle on to make the payload of the most recently failed request available for inspection. See Inspect Payload to Troubleshoot Closed Connections below.

Authentication method: Use the buttons to select one of these options:

- **Manual**: In the resulting **Auth token** field, enter the shared secret token to use when establishing a connection to a Splunk indexer.
- **Secret**: This option exposes an **Auth token (text secret)** drop-down, in which you can select a stored secret that references the auth token described above. A **Create** link is available to store a new, reusable secret.

Compression: This setting controls compression for Splunk-to-Splunk `v4` data. Select the **Compression** type from the drop-down list:

- `Disabled`: Compression is turned off. This is the default setting for both existing and new Destinations in Cribl Edge versions 4.11 and older. Connections will be closed if the server attempts to use compression.
- `Always`: Cribl Edge will always send compressed data, regardless of the receiver's compression configuration.

This setting is not available when **Max S2S version** is set to `v3`.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Inspect Payload to Troubleshoot Closed Connections

When a downstream receiver closes connections from this Destination (or just stops responding), inspecting the payload of the most recently failed request can help you find the cause. For example:

- Suppose you send an event whose size is larger than the downstream receiver can handle.
- Suppose you send an event that has a `number` field, but the value exceeds the highest number that the downstream receiver can handle.

When **Log failed requests to disk** is enabled, you can inspect the last failed request payload. Here is how:

1. In the Destination UI, navigate to the **Logs** tab.
2. Find a log entry with a `connection error` message.
3. Expand the log entry.
4. If the message includes the phrase `See payload file for more info`, note the path in the `file` field on the next line.

Now you have the path to the directory where Cribl Edge is storing the payload from the last failed request.

Notes about Forwarding to Splunk

- Data sent to Splunk is not compressed.
- The only `ack` from indexers that Cribl Edge listens for and acts upon is the shutdown signal described in Minimize in-flight data loss above.
- If events have a Cribl Edge internal field called `__criblMetrics`, they'll be forwarded to Splunk as metric events.
- If events do **not** have a `_raw` field, they'll be serialized to JSON prior to sending to Splunk.
- See Splunk's documentation on editing `fields.conf` to ensure the visibility of index-time fields sent to Splunk by Cribl Edge.

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

Cribl University offers an Advanced Troubleshooting > Destination Integrations: Splunk Cloud short course. To follow the direct course link, first log into your Cribl University account. (To create an account, select the **Sign up** link. You'll need to select through a short **Terms & Conditions** presentation, with chill music, before proceeding to courses - but Cribl's training is always free of charge.) Once logged in, check out other useful Advanced Troubleshooting short courses and Troubleshooting Criblets.

12.3.11.3. Splunk Load Balanced Destination

The **Splunk Load Balanced** Destination can load-balance the data it streams to multiple Splunk receivers. Downstream Splunk instances receive the data cooked and parsed.



Type: Streaming | TLS Support: Configurable | PQ Support: Yes

Looking for a quick way to switch all of your S2S Destinations to Splunk HEC Destinations? Follow our how-to guide: Switch Cribl Stream Destinations from S2S to Splunk HEC.

Splunk Cloud Platform

To establish secure connections to Splunk Cloud, you'll need to configure SSL settings using the private and public keys from the Splunk Cloud Universal Forwarder credentials package.

For details on which Destination to use for Splunk Cloud, see Splunk Cloud Platform and BYOL Integrations. The Splunk HEC Destination is recommended for most use cases.

Configure Cribl Edge to Load-Balance to Multiple Splunk Destinations

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this Splunk LB Destination definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
    - **Description**: Optionally, enter a description.
    - **Indexer discovery**: When enabled, Cribl Edge will automatically discover the indexers in an indexer clustering environment. </

This hides both **Exclude current host IPs** and the **Destinations** section, and displays the following fields:

- **Site**: Clustering site whose indexers need to be discovered. In the case of a single site cluster, `default` is the default entry.
- **Cluster Manager URI**: Full URI of Splunk cluster manager, in the format: `scheme://host:port`. (Edge Nodes normally access the cluster manager on **port 8089** to get the list of currently online indexers.)
- **Refresh period**: Time interval (in seconds) between two consecutive fetches of the indexer list from the cluster manager. Defaults to `300` seconds (five minutes).
- **Validate cluster manager certificates**: If toggled on (default), Cribl Edge is configured to reject any cluster manager certificate during indexer discovery that is not authorized by your system's Certificate Authority (CA). Cribl strongly recommends the default setting unless you must allow untrusted (for example, self-signed) certificates.

3. Under **Authentication tokens**, select one of the **Authentication method** options for authenticating to the cluster manager for indexer discovery:
   - **Manual**: In the resulting **Auth token** field, enter the required token.
   - **Secret**: This option exposes an **Auth token (text secret)** drop-down, from which you can select a stored secret that references the auth token. A **Create** link is available to store a new, reusable secret.

     If you use multiple cluster managers, you can add multiple tokens to cycle through them if a connection to the primary cluster manager fails. Select **Add token** to configure another authentication token.



Each Worker Process performs its own indexer discovery according to the above settings.

4. The **Destinations** section appears only when **Indexer discovery** is toggled off (default). Here, you specify a known set of Splunk receivers on which to load-balance data. Then you assign a weight to Cribl Edge's connection to each receiver. The higher the weight assigned to a given connection, the more traffic Cribl Edge will attempt to allocate to that connection. Click **Add Destination** to specify more receivers on new rows. Each row provides the following fields:
   - **Address**: Hostname or IP address of the receiver. If you enter a hostname, Cribl Edge tries to resolve it to its IP address or addresses. If a hostname has multiple A records - that is, if multiple receivers are behind it - Cribl Edge will assign the weight you assigned to the host to all of its IP addresses.
   - **Port**: Port number to send data to.
   - **TLS**: Whether to inherit TLS configs from group setting, or disable TLS. Defaults to `inherit`.

- **TLS servername**: Servername to use if establishing a TLS connection. If not specified, defaults to the connection host (if not an IP). Otherwise, uses the global TLS settings.
- **Load weight**: Set the relative traffic-handling capability for each connection by assigning a weight greater than `0`. For details, see Destinations below.

5. Next, you can configure the following **Optional Settings**:
   - **Exclude current host IPs**: Enable if you want to exclude all the current host's IP addresses from the list of resolved hostnames.
   - **Backpressure behavior**: Select whether to block, drop, or queue events when all receivers in this group are exerting backpressure. (Causes might include a broken or denied connection, or a rate limiter.) Defaults to `Block`. When toggled to `Persistent Queue`, adds the Persistent Queue Settings section (left tab) to the modal.
   - **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
6. Optionally, you can adjust the Persistent Queue, TLS, Processing, Timeout, and Advanced settings outlined in the sections below.
7. Select **Save**, then **Commit & Deploy**.

Destinations

- For best results, the weights for all connections should have the same order of magnitude. For example, if you have three connections, weights of `10`, `50`, and `40` would work; so would weights of `1`, `5`, and `4` - but weights like `10`, `120`, and `9` mix different orders of magnitude and should be avoided.
- You can add connections and keep them unused except for special circumstances such as testing. Just assign these connections a weight of `0` and Cribl Edge will ignore them. When you want to use such a "parked" connection, assign it a weight greater than `0`.

The final column provides an `X` button to delete any row from the **Destinations** table.



For details on configuring all load balancing settings, see About Load Balancing.

Enabling Cluster Manager Authentication

To enable token authentication on the Splunk cluster manager, you can find complete instructions in Splunk's Enable or Disable Token Authentication documentation. This option requires Splunk 7.3.0 or higher, and requires the following capabilites: `list_indexer_cluster` and `list_indexerdiscovery`.

For details on creating the token, see Splunk's Create Authentication Tokens topic - especially its section on how to Configure Token Expiry and "Not Before" Settings.



Be sure to give the token an **Expiration** setting well in the future, whether you use **Relative Time** or **Absolute Time**. Otherwise, the token will inherit Splunk's default expiration time of `+30d` (30 days in the future), which will cause indexer discovery to fail.

If you have a failover site configured on Splunk's cluster manager, Cribl respects this configuration, and forwards the data to the failover site in case of site failure.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.

Mode	Description
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5` GB. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.



Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

TLS Settings (Client Side)

Enabled: Defaults to toggled off. When toggled on:

Validate server certs: Reject any server (indexer) certificate that is not authorized by a CA in the **CA certificate path** or by another trusted CA (such as the CA for your system). Defaults to toggled on.

Server name (SNI): Server name for the SNI (Server Name Indication) TLS extension. This must be a hostname, not an IP address.

Minimum TLS version: Optionally, select the minimum TLS version to use when connecting.

Maximum TLS version: Optionally, select the maximum TLS version to use when connecting.

Certificate: Select an existing certificate name from the drop-down or add a new certificate by selecting **Create**. See Add a TLS Certificate to a Fleet for details.

CA certificate path: Path on client containing CA certificates (in PEM format) to use to verify the server's cert. Path can reference `$ENV_VARS`.

Private key path (mutual auth): Path on client containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Certificate path (mutual auth): Path on client containing certificates in (PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Passphrase: Passphrase to use to decrypt private key. Often the value of the `sslPassword` or similar parameter in the `outputs.conf` or `server.conf` file.



Single PEM File

> If you have a **single** `.pem` file containing `cacert`, `key`, and `cert` sections, enter this file's path in all of these fields above: **CA certificate path**, **Private key path (mutual auth)**, and **Certificate path (mutual auth)**.

Timeout Settings

- **Connection timeout**: Amount of time (in milliseconds) to wait for the connection to establish, before retrying. Defaults to `10000` ms.
- **Write timeout**: Amount of time (in milliseconds) to wait for a write to complete, before assuming connection is dead. Defaults to `60000` ms.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Advanced Settings

Output multiple metrics: Toggle on to output multiple-measurement metric data points. (Supported in Splunk 8.0 and above, this format enables sending multiple metrics in a single event, improving the efficiency of your Splunk capacity.)

Minimize in-flight data loss: If toggled on (default), Cribl Edge will check whether the indexer is shutting down, and if so, will stop sending data. This helps minimize data loss during shutdown. (Note that Splunk logs will indicate that the Cribl app has set `UseAck` to `true`, even though Cribl does not enable full `UseAck` behavior.) If toggled off, exposes the following alternative option:

Failed health check limit: Displayed (and set to `1` by default) only if **Minimize in-flight data loss** is disabled. This option sends periodic requests to Splunk once per minute, to verify that the Splunk endpoint is still alive and can receive data. Its value governs how many failed requests Cribl Edge will allow before closing this connection.

A low threshold value improves connections' resilience, but by proliferating connections, this can complicate troubleshooting. Set to `0` to disable health checks entirely - here, if the connection to Splunk is forcibly closed, you risk some data loss.

Max S2S version: The highest version of the Splunk-to-Splunk protocol to expose during handshake. Defaults to `v4`; `v3` is also available.

DNS resolution period (seconds): Re-resolve any hostnames after each interval of this many seconds, and pick up destinations from A records. Defaults to `600` seconds.

Load balance stats period (seconds): How far back in time to keep traffic stats for load balancing purposes. Defaults to `300` seconds (five minutes).

Connection limit: Constrains the number of concurrent indexer connections, per Worker Process, to limit memory utilization. If set to a number > `0`, then on every DNS resolution period (or indexer discovery), Cribl Edge will randomly select this subset of discovered IPs to connect to. Cribl Edge will rotate IPs in future resolution periods - monitoring weight and historical data, to ensure fair load balancing of events among IPs.

Nested field serialization: Specifies whether and how to serialize nested fields into index-time fields. Select `None` (the default) or `JSON`.

Authentication method: Use the buttons to select one of these options:

- **Manual**: In the resulting **Auth token** field, enter the shared secret token to use when establishing a connection to a Splunk indexer.
- **Secret**: This option exposes an **Auth token (text secret)** drop-down, in which you can select a stored secret that references the auth token described above. A **Create** link is available to store a new, reusable secret.

Compression: This setting controls compression for Splunk-to-Splunk `v4` data. Select the **Compression** type from the drop-down list:

- `Disabled`: Compression is turned off. This is the default setting for both existing and new Destinations in Cribl Edge versions 4.11 and older. Connections will be closed if the server attempts to use compression.
- `Always`: Cribl Edge will always send compressed data, regardless of the receiver's compression configuration.

This setting is not available when **Max S2S version** is set to `v3`.

Log failed requests to disk: Toggle on to make the payload of the most recently failed request available for inspection. See Inspect Payload to Troubleshoot Closed Connections below.

Endpoint health fluctuation time allowance (ms): How long (in milliseconds) each receiver endpoint can report blocked, before this Destination as a whole reports unhealthy, blocking senders. (Grace period for transient fluctuations.) Use `0` to disable the allowance; default is `100` ms; maximum allowed value is `60000` ms (1 minute).

Throttling: Throttle rate, in bytes per second. Multiple byte units such as KB, MB, GB, and so forth, are also allowed - for example, `42 MB`. A value of `0` (the default) indicates no throttling. When throttling is engaged, excess data will be dropped only if **Backpressure behavior** is set to **Drop events**. (Data will be blocked for all other **Backpressure behavior** settings.)

- The throttling rate is applied per Worker Process, per configured endpoint. For example, if you have two Worker Processes, each with two endpoints, and you set the throttling rate to `3 MB`, the total potential throughput rate is 12 MB per second: up to 3 MB per second for each of the four endpoints. The **Connection limit** setting is also applied per Worker Process. To continue the example, if you set **Connection limit** to `1`, the total potential throughput is 6 MB per second, even though each Worker Process has two configured endpoints.
- The throttling rate is applied after compression.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Inspect Payload to Troubleshoot Closed Connections

When a downstream receiver closes connections from this Destination (or just stops responding), inspecting the payload of the most recently failed request can help you find the cause. For example:

- Suppose you send an event whose size is larger than the downstream receiver can handle.
- Suppose you send an event that has a `number` field, but the value exceeds the highest number that the downstream receiver can handle.

When **Log failed requests to disk** is enabled, you can inspect the last failed request payload. Here is how:

1. In the Destination UI, navigate to the **Logs** tab.
2. Find a log entry with a `connection error` message.
3. Expand the log entry.
4. If the message includes the phrase `See payload file for more info`, note the path in the `file` field on the next line.

Now you have the path to the directory where Cribl Edge is storing the payload from the last failed request.

SSL Configuration for Splunk Cloud

To connect to Splunk Cloud, you will need to extract the private and public keys from the Splunk-provided Splunk Cloud Universal Forwarder credentials package. You will also need to reference the CA Certificate located in the same package.

You can reuse many of the settings in this Splunk Cloud package to set up Splunk Cloud Destinations. Use the following steps:

Step 1. Extract the `splunkclouduf.spl` package on the Cribl Edge instance that you will be connecting to Splunk Cloud. You will have a folder that looks something like this:

```
100_my-splunk-cloud_splunkcloud
        /default/
                outputs.conf
                limits.conf
                your-splunk-cloud_server.pem
                your-splunk-cloud_cacert.pem
```

Copy

Step 2. (optional) Test connectivity to Splunk Cloud, using the Root CA certificate:

```
echo | openssl s_client -CAfile 100_<your-splunk-cloud>_splunkcloud/default/my-splunk
```

Copy

To test the connection, you can use any of the URLs listed in the `[tcpout:splunkcloud]` stanza's `outputs.conf` section.

You can simplify Steps 3 and 4 below by dragging and dropping (or uploading) the `.pem` files into Cribl Edge's **New Certificates** modal. See Configure TLS for API and UI Access.

Step 3. Extract the private key from the Splunk Cloud certificate. At the prompt, you will need the `sslPassword` value from the `outputs.conf`. Using Elliptic Curve keys:

```
openssl ec -in 100_<your-splunk-cloud>_splunkcloud/default/<your-splunk-cloud>_server
```

Copy

If you are using RSA keys, instead use:

```
openssl rsa -in 100_<your-splunk-cloud>_splunkcloud/default/<your-splunk-cloud>_serve
```

Copy

Step 4. Extract the public key for the Server Certificate:

```
openssl x509 -in 100_<your-splunk-cloud>_splunkcloud/default/<your-splunk-cloud>_serv
```

Copy

Step 5. In the Splunk Load Balanced Destination's TLS Settings (Client Side) section, enter the following:

- CA Certificate Path: Path to `<your-splunk-cloud>_cacert.pem`.
- Private Key Path (mutual auth): Path to `private.pem` (Step 3 above).
- Certificate Path (mutual auth): Path to `server.pem` (Step 4 above).

In a distributed deployment, enter this Destination configuration on each Worker Group/Fleet that forwards to Splunk Cloud. Then commit and deploy your changes.

Step 6. In a distributed deployment, enable Worker UI access, and verify that the Certificate files have been distributed to individual workers. If they are not present, copy the Certificate files to the Workers, using exactly the same paths you used at the Group level.

Notes About Forwarding to Splunk

- Data sent to Splunk is **not** compressed.
- The only `ack` from indexers that Cribl Edge listens for and acts upon is the shutdown signal described in Minimize in-flight data loss above.
- If events have a Cribl Edge internal field called `__criblMetrics`, they'll be forwarded to Splunk as metric events.
- If events do not have a `_raw` field, they'll be serialized to JSON prior to sending to Splunk.
- You can copy and paste the Splunk Cloud servers from the `[tcpout:splunkcloud]` stanza into **General Settings** > **Destinations**. For example, from the example stanza below, you would copy only the bolded contents:

  ~~[tcpout:splunkcloud]~~
  ~~server =~~ **inputs1.your-splunk-cloud.splunkcloud.com:9997, inputs2.your-splunk-cloud.splunkcloud.com:9997, inputs3.your-splunk-cloud.splunkcloud.com:9997, inputs4.your-splunk-**

cloud.splunkcloud.com:9997, inputs5.your-splunk-cloud.splunkcloud.com:9997, inputs6.your-splunk-cloud.splunkcloud.com:9997, inputs7.your-splunk-cloud.splunkcloud.com:9997, inputs8.your-splunk-cloud.splunkcloud.com:9997, inputs9.your-splunk-cloud.splunkcloud.com:9997, inputs10.your-splunk-cloud.splunkcloud.com:9997, inputs11.your-splunk-cloud.splunkcloud.com:9997, inputs12.your-splunk-cloud.splunkcloud.com:9997, inputs13.your-splunk-cloud.splunkcloud.com:9997, inputs14.your-splunk-cloud.splunkcloud.com:9997, inputs15.your-splunk-cloud.splunkcloud.com:9997
~~compressed = false~~

- From `limits.conf`, copy the `[thruput]` value, then paste it into **Advanced Settings** > **Throttling**.
- If you enable both (1) indexer discovery and (2) TLS with certificate validation, you see might errors. This happens because the IPs discovered (by default) for the indexers will not match the fully qualified domain names (FQDNs) referenced by the indexer certificates. This causes the TLS handshake to fail.
    - As a workaround, use `server.conf` on each indexer, setting `register_forwarder_address = <your.idx.fqdn>`. Cribl Edge will now get that value, and the certificates will match.
- See Splunk's documentation on editing `fields.conf` to ensure the visibility of index-time fields sent to Splunk by Cribl Edge.

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

Cribl University's Advanced Troubleshooting short courses include Destination Integrations: Splunk LB and Destination Integrations: Splunk Cloud. To follow these direct course links, first log into your Cribl University account. (To create an account, select the **Sign up** link. You'll need to select through a short

12.3.12. CrowdStrike Falcon LogScale Destination

The **CrowdStrike Falcon LogScale** Destination can stream data to a LogScale HEC (HTTP Event Collector) in JSON or Raw format.

Type: Streaming | TLS Support: Configurable | PQ Support: Yes

(In Cribl Edge 3.5.x, this Destination was labeled **Humio HEC**. Some links from this page might still lead to "Humio"-branded resources that CrowdStrike has not renamed.)

Recommendations

To load-balance to customer-managed LogScale receivers, Cribl recommends placing a load balancer in front of cluster nodes, following LogScale Manual Cluster Deployment recommendations. If you are using LogScale Cloud, it provides its own load balancing.

We recommend sending events with the `sourceType` field set to a LogScale parser. This tells LogScale which parser to use to extract fields (for example, `"sourceType":"json"`).

If LogScale cannot match the `sourceType` value to a parser, it will use the `kv` parser, and you will get an error that LogScale could not resolve the specified parser. Alternatively, you can assign a parser to the ingest token that you use to authenticate this Destination.

The `fields` element does not support Nested JSON. Any nested elements will be dropped.

Configure Cribl Edge to Output to CrowdStrike Falcon LogScale Destinations

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.

2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this CrowdStrike Falcon LogScale definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
    - **Description**: Optionally, enter a description.
    - **LogScale endpoint**: URL of a CrowdStrike Falcon LogScale endpoint to send events to (for example, `https://cloud.us.humio.com:443/api/v1/ingest/hec`).

      JSON-formatted events normally go to `/api/v1/ingest/hec` or `/services/collector`.

      Raw (simple line-delimited) events normally go to `/api/v1/ingest/hec/raw` or `/services/collector/raw`.
    - **Request format**: Select how you want events formatted, either `JSON` or `Raw`. Make sure your selection here matches the **LogScale endpoint** you specify above.
3. Under **Authentication**, select an **Authentication method** from the dropdown:
    - **Manual**: Displays a **LogScale Auth token** field for you to enter your CrowdStrike Falcon LogScale HEC API token.
    - **Secret**: Displays a **LogScale token (text secret)** drop-down, in which you can select a stored secret that references the API token described above. A **Create** link is available to store a new, reusable secret.
4. Next, you can configure the following **Optional Settings**:
    - **Backpressure behavior**: Select whether to block, drop, or queue events when all receivers are exerting backpressure. (Causes might include a broken or denied connection, or a rate limiter.) Defaults to `Block`.
    - **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
5. Optionally, you can adjust the Persistent Queue, Processing, Retries, and Advanced settings outlined in the sections below.
6. Select **Save**, then **Commit & Deploy**.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5 GB`. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.



Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.

- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Retries

Honor Retry-After header: Toggle on to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream/Edge limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. Any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. Toggle off to ignore all `Retry-After` headers.

Default Retry Behavior: By default, Cribl Stream/Edge applies predefined retry strategies for certain failed requests to the Destination using an exponential backoff algorithm. You can override these settings by manually configuring the retry behavior for a particular error code.



Destinations created before the 4.16.0 release do not support the default `4xx` series retry behavior. If your Destination was created prior to this release and you'd like it to support the behavior, reconfigure the Destination.

To retry requests, Cribl Stream/Edge applies the following rules by default:

Status Code	Action
Any in the `1xx`, `3xx` series	Drop the request
`401`, `403`, `408`, `429` errors	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached
Any other `4xx` series	Drop the request
Any in the `5xx` series	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached

By default, any `429 (Too Many Requests)` response that is not listed in the retry settings is treated as a non-retryable `4xx` error and dropped. If you want `429` responses retried, ensure `429` is included in the **Settings for failed HTTP requests**.

Settings for failed HTTP requests: You can also manually configure Cribl Stream/Edge to automatically retry requests that receive particular HTTP response status codes.

Upon receiving a response code that's on the list, Cribl Stream/Edge first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream/Edge continues retrying the request without increasing the time between retries any further.

If the sender (which manages the connection to the Destination) is at capacity, it will not accept any incoming events. These incoming events originate internally from a previous stage of the data flow when Destinations send outbound requests to their respective external services, and they include retry requests and new requests. Any events that were already in transit when the sender reached capacity will continue to be processed downstream.

Sender capacity is freed up when an outgoing request succeeds or encounters a non-retryable error. When the sender has available capacity again, it will resume accepting incoming events. This capacity management is influenced by the number of active connections and configured limits, such as concurrency and buffer sizes. If a Pipeline sends events faster than the Destination can process, the buffers may fill up, leading to backpressure and `Sender at capacity` warnings. This backpressure prevents the sender from accepting additional requests until capacity is restored.

For each response code you want to add to the list, select **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: Toggle on to automatically retry requests that have timed out and display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Defaults to toggled on to reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (for example, the system's CA).

Round-robin DNS: Defaults to toggled on to enable round-robin DNS lookup across multiple IP addresses, IPv4 and IPv6. When a DNS server resolves a Fully Qualified Domain Name (FQDN) to multiple IP addresses, Cribl Edge will sequentially use each address in the order they are returned by the DNS server for subsequent connection attempts.

Compress: Defaults to toggled on to compress the payload body before sending.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests per Worker Process. When Cribl Edge hits this limit, it begins throttling traffic to the downstream service. Defaults to `5`. Minimum: `1`. Maximum: `32`. Each request can potentially hit a different HEC receiver.

Body size limit (KB): Maximum size, in KB, of the request body. Defaults to `4096`. Lowering the size can potentially result in more parallel requests and also cause outbound requests to be made sooner.

Events-per-request limit: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values can cause the payload size to be smaller than the configured **Body size limit**. Defaults to `1`.



- Retries happen on this flush interval.
- Any HTTP response code in the `2xx` range is considered success.
- Any response code in the `5xx` range is considered a retryable error, which will not trigger Persistent Queue (PQ) usage.
- Any other response code will trigger PQ (if PQ is configured as the Backpressure behavior).

Extra HTTP headers: Click **Add Header** to add **Name/Value** pairs to pass as additional HTTP headers. Values will be sent encrypted.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Cribl Edge will redact all headers, except non-sensitive headers that you declare below in **Safe headers**.

Safe headers: Add headers that you want to declare as safe to log in plaintext. (Sensitive headers such as `authorization` will always be redacted, even if listed here.) Use a tab or hard return to separate header names.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Notes on HTTP-Based Outputs

- To proxy outbound HTTP/S requests, see System Proxy Configuration.
- Cribl Edge will attempt to use keepalives to reuse a connection for multiple requests. After two minutes of the first use, the connection will be thrown away, and a new connection will be reattempted. This is to prevent sticking to a particular Destination when there is a constant flow of events.
- If the server does not support keepalives - or if the server closes a pooled connection while idle - a new connection will be established for the next request.
- Cribl Edge will pick the first IP in the list for use in the next connection. Enable Round-robin DNS to better balance distribution of events between CrowdStrike Falcon LogScale servers.

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

12.3.13. ClickHouse Destination

The ClickHouse Destination enables seamless integration with ClickHouse. This Destination routes observability data, including logs, metrics, and traces, to ClickHouse for enhanced monitoring and analysis.

Data is formatted and transmitted via HTTP requests, offering both synchronous and asynchronous insert options for flexibility. Authentication is secured with Basic or SSL User Certificate methods, while robust retry settings ensure reliable data delivery. Flexible data mapping, including automatic and manual field-to-column mapping, guarantees structured ingestion. To optimize performance, the integration supports batching and asynchronous inserts, leveraging ClickHouse's capabilities for efficient data delivery.

To get started, connecting to ClickHouse is simple. Grab your ClickHouse instance's HTTP interface URL, database name, and table name, then add these details to your Destination settings. Need more detailed instructions or configuration options? See below!



Type: Streaming | TLS Support: Configurable | PQ Support: Yes

Configure a ClickHouse Destination

In Cribl Edge, set up a ClickHouse Destination.

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing**.
    - To configure via the Routes, select **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the Destination modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this output definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**. A **Description** is optional.
    - **URL**: The URL of your ClickHouse instance's HTTP interface. This should include the protocol (http or https) and the port number if it's not the default. Example: `http://localhost:8123`.
    - **ClickHouse database**: Enter the name of the ClickHouse database where the target table resides. Ensure that the database exists and the authenticated user has the necessary permissions to write to it.

- **ClickHouse table**: Enter the name of the target table. The name can contain letters (`A-Z`, `a-z`), numbers (`0-9`), and underscores `_`, and must start with either a letter or underscore. The table must exist and have a schema compatible with the data being sent.
- **Format**: Choose the format for events sent to ClickHouse. Choose between:
  - **JSONCompactEachRowWithNames** (default): Optimized for efficiency by minimizing data size, making it ideal for transmitting data to ClickHouse over networks where bandwidth is a concern.
  - **JSONEachRow**: Standard JSON format with extra spacing, useful for debugging or human inspection. If the time spent reformatting data into the compact format exceeds the cost of transmitting more data, `JSONEachRow` may be preferable. For example, in scenarios where the ClickHouse instance runs on the same machine as Cribl, the network cost is negligible, and the time spent reformatting doesn't provide significant benefits.
- **Authentication type**: ClickHouse supports several authentication methods for its HTTP interface.
  - **None**: Don't use authentication.
  - **Basic**: Displays **Username** and **Password** fields for you to enter HTTP Basic authentication credentials.
  - **Basic (credentials secret)**: This option exposes a **Credentials secret** drop-down to select a stored text secret that contains your username and password credentials referenced by a secret. A **Create** link is available to store a new, reusable secret.
  - **SSL User Certificate**: Use to securely connect to ClickHouse. Provide the username for certificate authentication. This method ensures a higher level of security by using SSL certificates to verify the identity of the client.
- **Mapping**: Determines how fields from Cribl events are mapped to ClickHouse columns. Options include `Automatic` and `Custom`.
  - **Automatic**: Uses default mappings.
    - **Exclude fields**: A list of fields to exclude from the data sent to ClickHouse. This can help reduce the size of the data and avoid sending unnecessary information.
  - **Custom**: Manually specify column mappings. Select **Retrieve table columns** to connect to a publicly accessible ClickHouse instance, authenticate using the provided credentials, and fetch the table schema. The retrieved column names and types will be automatically populated in the **Column Mapping** table, allowing you to focus on mapping the appropriate data values.
    - The **Column Mapping** table allows you to define how event data fields are mapped to ClickHouse table columns. This is essential for ensuring that data is correctly ingested into ClickHouse with the appropriate structure and types. See Data Types in ClickHouse for details.

        - **Column name**: The name of the column in the ClickHouse table. This should match the column name defined in the ClickHouse schema.
        - **Column type**: The data type of the column in the ClickHouse table (such as, `String`, `Int32`, `DateTime`). This ensures that the data is stored in the correct format.
        - **Column value**: A JavaScript expression that defines how to extract or transform the event data to fit into the specified column. This allows for flexible data manipulation and transformation before ingestion.
    - **Async**: The **Async inserts** toggle enables asynchronous inserts, allowing ClickHouse to buffer data in memory before writing it to disk. This can improve performance but may risk data loss if the server crashes before the data is written.
        - **Wait for async inserts** waits for the ClickHouse server to confirm that the data has been fully written to disk before proceeding. This ensures a higher level of data integrity by reducing the risk of data loss if the server encounters an issue before full insertion. This setting balances throughput and data assurance, making it suitable for high-volume logging where occasional data loss is acceptable.
    - **Optional Settings**:
        - **Backpressure behavior**: Whether to `Block`, `Drop`, or `Persistent Queue` events when all receivers are exerting backpressure.
        - **Tags**: Optionally, add tags that you can use to filter and group Destinations in Cribl Edge's **Manage Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
3. If you're using TLS, select **TLS Settings (Client Side)** on the left tabs and toggle **Use TLS** on. See the TLS Settings section for details. Default is toggled off.
4. The **Processing Settings**, **Retries**, and **Advanced Settings** provide granular control over how data is handled and transmitted.
    - Processing Settings allow you to define pre-processing and post-processing Pipelines, ensuring that events conform to the relevant Protobuf specifications for traces, metrics, and logs. This includes adding metadata, applying transformations, and conditioning the data before it is sent to the destination.
    - Retries allow you to specify how the system should handle different HTTP response status codes and timeouts, ensuring reliable data delivery.
    - Advanced Settings offer additional configuration options that enhance the flexibility and performance of data transmission.
5. Select **Save**, then **Commit & Deploy**.
6. Use the **Test** tab in the Destination's configuration modal to validate the configuration and connectivity of your Destination.

TLS Settings (Client Side)

Server name (SNI): Server name for the SNI (Server Name Indication) TLS extension. This must be a host name, not an IP address.

Minimum TLS version: Optionally, select the minimum TLS version to use when connecting.

Maximum TLS version: Optionally, select the maximum TLS version to use when connecting.

Certificate: Select an existing certificate name from the drop-down or add a new certificate by selecting **Create**. See Add a TLS Certificate to a Fleet for details.

CA certificate path: Path on client containing CA certificates (in PEM format) to use to verify the server's cert. Path can reference `$ENV_VARS`.

Private key path (mutual auth): Path on client containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Certificate path (mutual auth): Path on client containing certificates in (PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Passphrase: Passphrase to use to decrypt private key.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

> On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.
>
> This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a

> hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

> If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5 GB`. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.



Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output - both metric events, as dimensions; and log events, as labels. Supports wildcards.

By default, includes `cribl_pipe` (Cribl Edge Pipeline that processed the event).

Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Retries

Honor Retry-After header: Toggle on to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream/Edge limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. Any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. Toggle off to ignore all `Retry-After` headers.

Default Retry Behavior: By default, Cribl Stream/Edge applies predefined retry strategies for certain failed requests to the Destination using an exponential backoff algorithm. You can override these settings by manually configuring the retry behavior for a particular error code.



Destinations created before the 4.16.0 release do not support the default `4xx` series retry behavior. If your Destination was created prior to this release and you'd like it to support the behavior, reconfigure the Destination.

To retry requests, Cribl Stream/Edge applies the following rules by default:

Status Code	Action
Any in the `1xx`, `3xx` series	Drop the request
`401`, `403`, `408`, `429` errors	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached
Any other `4xx` series	Drop the request
Any in the `5xx` series	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached

By default, any `429 (Too Many Requests)` response that is not listed in the retry settings is treated as a non-retryable `4xx` error and dropped. If you want `429` responses retried, ensure `429` is included in the **Settings for failed HTTP requests**.

Settings for failed HTTP requests: You can also manually configure Cribl Stream/Edge to automatically retry requests that receive particular HTTP response status codes.

Upon receiving a response code that's on the list, Cribl Stream/Edge first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream/Edge continues retrying the request without increasing the time between retries any further.

If the sender (which manages the connection to the Destination) is at capacity, it will not accept any incoming events. These incoming events originate internally from a previous stage of the data flow when Destinations send outbound requests to their respective external services, and they include retry requests and new

requests. Any events that were already in transit when the sender reached capacity will continue to be processed downstream.

Sender capacity is freed up when an outgoing request succeeds or encounters a non-retryable error. When the sender has available capacity again, it will resume accepting incoming events. This capacity management is influenced by the number of active connections and configured limits, such as concurrency and buffer sizes. If a Pipeline sends events faster than the Destination can process, the buffers may fill up, leading to backpressure and `Sender at capacity` warnings. This backpressure prevents the sender from accepting additional requests until capacity is restored.

For each response code you want to add to the list, select **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: Toggle on to automatically retry requests that have timed out and display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Reject certificates that are **not** authorized by a CA in the **CA certificate path**, nor by another trusted CA (for example, the system's CA). Defaults to toggled on.

Round-robin DNS: Toggle on to enable round-robin DNS lookup across multiple IP addresses, IPv4 and IPv6. When a DNS server resolves a Fully Qualified Domain Name (FQDN) to multiple IP addresses, Cribl Edge will sequentially use each address in the order they are returned by the DNS server for subsequent connection attempts.

Compress: Toggle on (default) to compress the payload body before sending.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests per Worker Process. When Cribl Edge hits this limit, it begins throttling traffic to the downstream service. Defaults to `5`. Minimum: `1`. Maximum: `32`.

Body size limit (KB): Maximum size of the request body before compression. Defaults to `4096` KB. You can set this limit to as high as 10 MB (`10240` KB). Be aware that high values can cause high memory usage per Worker Node, especially if a dynamically constructed URL (see Internal Fields) causes this Destination to send events to more than one URL. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Body size limit** value until downstream receivers reliably accept all events.

Events-per-request limit: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Extra HTTP headers: Name-value pairs to pass to all events as additional HTTP headers. Values will be sent encrypted. You can also add headers dynamically on a per-event basis in the `__headers` field. See Internal Fields below.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Cribl Edge will redact all headers, except non-sensitive headers that you declare below in **Safe headers**.

Safe headers: Add headers to declare them as safe to log in plaintext. (Sensitive headers such as `authorization` will always be redacted, even if listed here.) Use a tab or hard return to separate header names.

Log last schema mismatch: Toggle on to enable logging of the most recent event that fails to match the ClickHouse table schema. When enabled, an entry with the message `Failed to serialize event. See payload file for more info.` is written to the system log. This log entry points you to the temporary output file `last-failed-format-event.json`, which contains the data for the most recent failed event.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Troubleshoot

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

12.3.14. Cloudflare R2 Destination

The Cloudflare R2 Destination sends data to Cloudflare R2 object storage using an S3-compatible API.

Type: Non-Streaming | TLS Support: Configurable | PQ Support: No

Configure a Cloudflare R2 Destination

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
    - **Description**: Optionally, enter a description.
    - **R2 endpoint**: Cloudflare R2 service URL. For example: `https://<ACCOUNT_ID>.r2.cloudflarestorage.com`.
    - **R2 bucket name**: Name of the R2 bucket. This value can be a constant, or a JavaScript expression that will be evaluated only at init time. For example, referencing a Global Variable: `myBucket-${C.vars.myVar}`. Ensure that the bucket already exists, otherwise R2 will generate "bucket does not exist" errors.

      Event-level variables are not available for JavaScript expressions. This is because the bucket name is evaluated only at Destination initialization. If you want to use event-level variables in file paths specify them in the **Partitioning Expression** field, because this is evaluated for each file.

    - **Staging location**: Filesystem location in which to locally buffer files before compressing and sending. Cribl recommends that this location be stable and high-performance.

The **Staging location** field is not displayed or available on Cribl.Cloud-managed Edge Nodes.

- **Data format**: The output data format defaults to `JSON`. `Raw` and `Parquet` are also available. Selecting `Parquet` (supported only on Linux, not Windows) exposes a **Parquet Settings** left tab, where you **must** configure certain options in order to export data in Parquet format.

3. Next, you can configure the following **Optional Settings**:
   - **Key prefix**: A constant or JavaScript expression that is prepended to each object key (file name) when writing data to the bucket. The key prefix is evaluated per file and can use global variables or environment variables, but not event-level fields. This enables logical organization of objects within the bucket, simulating directory structures using forward slashes `/`.
   - **Partitioning expression**: JavaScript expression that defines how files are partitioned and organized. Default is date-based. If blank, Cribl Edge will fall back to the event's `__partition` field value (if present); or otherwise to the root directory of the **Output Location** and **Staging Location**.



Cribl Edge's internal `__partition` field can be populated in multiple ways. The precedence order is: explicit **Partitioning expression** value -> `${host}/${sourcetype}` (default) **Partitioning expression** value -> user-defined `event.__partition`, set with an Eval Function (takes effect only where this **Partitioning expression** field is blank).

   - **Compression**: Data compression format used before sending. Defaults to `gzip` (recommended). This setting is not available when **Data format** is set to `Parquet`.
   - **File name prefix expression**: The output file name prefix. Must be a JavaScript expression (which can evaluate to a constant), enclosed in quotes or backticks. Defaults to `CriblOut`.
   - **File name suffix expression**: The output file name suffix. Must be a JavaScript expression (which can evaluate to a constant), enclosed in quotes or backticks. Defaults to `` `.${C.env["CRIBL_WORKER_ID"]}.${__format}${__compression === "gzip" ? ".gz" : ""}` ``, where `__format` can be `json` or `raw`, and `__compression` can be `none` or `gzip`.
   - **Backpressure behavior**: Select whether to block or drop events when all receivers are exerting backpressure. (Causes might include an accumulation of too many files needing to be closed.) Defaults to `Block`.
   - **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

4. Optionally, you can adjust the Authentication, Processing, Parquet, and Advanced settings outlined in the sections below.

5. Select **Save**, then **Commit & Deploy**.

Authentication

Use the **Authentication Method** drop-down to select either **Auto** or **Secret**:

Auto: This default option uses the AWS SDK for JavaScript to automatically obtain credentials in the following order of attempts:

1. **Environment Variables**: Loaded from environment variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`.
2. **IAM Identity Center (SSO)**: Loaded from AWS IAM Identity Center configuration.
3. **Shared Credentials File**: Loaded from the shared credentials file (`~/.aws/credentials`).
4. **IAM Roles for Amazon EC2/ECS**: Loaded from AWS Identity and Access Management (IAM) roles attached to an EC2 instance or ECS container.
5. **JSON File on Disk**: Loaded from a JSON file on disk.
6. **Other Credential-Provider Classes**: Other credential-provider classes provided by the AWS SDK for JavaScript.

The `Auto` method works both when running on AWS and in other environments where the necessary credentials are available through one of the above methods.



SSO Providers

When using the auto authentication method, you can leverage SSO providers like SAML and Okta to issue temporary credentials. These credentials should be set in the environment variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`. The AWS SDK will then use these environment variables to authenticate.

Secret: If not running on AWS, you can select this option to supply a stored secret that references an AWS access key and secret key. This option exposes a **Secret key pair** drop-down, in which you can select a stored secret that references the set of user-associated IAM credentials described above. A **Create** link is available to store a new, reusable secret.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.

- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Parquet Settings

To write out Parquet files, note that:

- On Linux, you can use the Cribl Edge CLI's `parquet` command to view a Parquet file, its metadata, or its schema.
- Cribl Edge Workers support Parquet only when running on Linux, not on Windows.
- See Parquet Schemas for pointers on how to avoid problems such as data mismatches.

Automatic schema: Toggle on to automatically generate a Parquet schema based on the events of each Parquet file that Cribl Edge writes. When toggled off (the default), exposes the following additional field:

- **Parquet schema**: Select a schema from the drop-down.



If you need to modify a schema or add a new one, follow the instructions in our Parquet Schemas topic. These steps will propagate the freshest schema back to this drop-down.

Parquet version: Determines which data types are supported, and how they are represented. Defaults to `2.6`; `2.4` and `1.0` are also available.

Data page version: Serialization format for data pages. Defaults to `V2`. If your toolchain includes a Parquet reader that does not support `V2`, use `V1`.

Group row limit: The number of rows that every group will contain. The final group can contain a smaller number of rows. Defaults to `10000`.

Page size: Set the target memory size for page segments. Generally, set lower values to improve reading speed, or set higher values to improve compression. Value must be a positive integer smaller than the **Row group size** value, with appropriate units. Defaults to `1 MB`.

Log invalid rows: Toggle on to output up to 20 unique rows that were skipped due to data format mismatch. Log level must be set to `debug` for output to be visible.

Write statistics: Leave toggled on (the default) if you have Parquet tools configured to view statistics - these profile an entire file in terms of minimum/maximum values within data, numbers of nulls, etc.

Write page indexes: Leave toggled on (the default) if your Parquet reader uses statistics from Page Indexes to enable page skipping. One Page Index contains statistics for one data page.

Write page checksum: Toggle on if you have configured Parquet tools to verify data integrity using the checksums of Parquet pages.

Metadata (optional): The metadata of files the Destination writes will include the properties you add here as key-value pairs. For example, one way to tag events as belonging to the OCSF category for security findings would be to set **Key** to `OCSF Event Class` and **Value** to `2001`.

Advanced Settings

File size limit (MB): Maximum uncompressed output file size. Files of this size will be closed and moved to final output location. Defaults to `32`.

File open time limit (sec): Maximum amount of time to write to a file. Files open for longer than this limit will be closed and moved to final output location. Defaults to `300`.

Idle time limit (sec): Maximum amount of time to keep inactive files open. Files open for longer than this limit will be closed and moved to final output location. Defaults to `30`.

Open file limit: Maximum number of files to keep open concurrently. When exceeded, the oldest open files will be closed and moved to final output location. Defaults to `100`.



Cribl Edge will close files when **either** of the `File size limit (MB)` or the `File open time limit (sec)` conditions is met.

Concurrent file parts limit: Maximum number of parts to upload in parallel per file. A value of `1` tells the Destination to send the whole file at once. When set to `2` or above, IAM permissions must include those required for multipart uploads. Defaults to `4`; highest allowed value is `10`.

Disk space protection: Specifies whether to `Block` (default) or `Drop` incoming events when the disk space falls below the globally defined **Min free disk space** amount.

Add Output ID: When toggled on (default), adds the **Output ID** field's value to the staging location's file path. This ensures that each Destination's logs will write to its own bucket.

Remove empty staging dirs: Toggle this on to delete empty staging directories after moving files. This prevents the proliferation of orphaned empty directories. When enabled, exposes this additional option:

- **Staging cleanup period**: How often (in seconds) to delete empty directories when **Remove empty staging dirs** is enabled. Defaults to `300` seconds (every 5 minutes). Minimum configurable interval is `10` seconds; maximum is `86400` seconds (every 24 hours).

Storage class: Select a storage class for uploaded objects. Defaults to `Standard`.

Server-side encryption: Server side encryption type for uploaded objects. Defaults to `none`.

Reuse connections: Whether to reuse connections between requests. Toggling on (default) can improve performance.

Reject unauthorized certificates: Whether to accept certificates that cannot be verified against a valid Certificate Authority (for example, self-signed certificates). Defaults to toggled on.

Verify if bucket exists: Toggle off if you can access files within the bucket, but not the bucket itself. This resolves errors of the form: `error initializing...Bucket does not exist`.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

IAM Permissions

The following permissions are always needed to write to an Amazon S3-compatible object store:

- `s3:ListBucket`
- `s3:GetBucketLocation`
- `s3:PutObject`

If your Destination needs to do multipart uploads to S3, two more permissions are needed:

- `kms:GenerateDataKey`
- `kms:Decrypt`

See the AWS documentation.

Internal Fields

Cribl Edge uses a set of internal fields to assist in forwarding data to a Destination.

Field for this Destination:

- `__partition`

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

Common Issues

"Bucket does not exist - self signed certificate"

Example Error Text:

```
message: Bucket does not exist - self signed certificate
stack: Error: self signed certificate
    at TLSSocket.onConnectSecure (node:_tls_wrap_1535:34)
    at TLSSocket.emit (node:events:513:28)
    at TLSSocket.emit (node:events:489:12)
    at TLSSocket._fini... Show more
tmpPath: /opt/cribl/s3_bucket/cribl/2023/03/28/CriblOut-XTNGh5.0.json.tmp
```

Copy

Can occur when:

- the Destination uses a self-signed cert that has not yet been trusted by Cribl Stream.
- the user does not have appropriate permissions to view the bucket.
- the authentication token generated by your computer has a timestamp that is out of sync with the server's time, resulting in an authentication failure.

Recommendation

Disable **Advanced Settings** > **Reject unauthorized certificates** or; get the sender's cert and add it to the `NODE_EXTRA_CA_CERTS` path for validation; or verify that the user has appropriate permissions.

12.3.15. Cortex XSIAM Destination

The **Cortex XSIAM** Destination allows you to stream event data to Palo Alto Networks' Cortex XSIAM (Extended Security Intelligence and Automation Management) for advanced threat detection and response.

Type: Streaming | TLS Support: Yes | PQ Support: Yes

XSIAM Data Requirements

Cribl Edge provides a generic **Palo Alto XSIAM** Pack designed to extract and map mandatory fields. Cribl Edge automatically validates and converts these fields into HTTP headers, ensuring compliance with XSIAM requirements. For details on using the Pack, see how to map data to the XSIAM Destination.

Optimized Data Delivery:

- **Automatic Format Parsing**: Detects event formats, including CEF, LEEF, Syslog, JSON, and Raw.
- **Batched Transmission**: To optimize data ingestion and reduce overhead, events are grouped into batches. Set the **Flush period** to control latency. Allowed values are `1` second or greater.
- **Aggregation**: Uses batched keys derived from the event format and system fields.
- **Payload Size**: Enforces a 5 MB limit per individual event and a 9.5 MB limit per batch. Warnings are logged if data exceeds these limits.
- **Rate Limiting**: There's a default request rate limit of `400` requests per second. Customizable with the **Throttle request rate limit** setting to align with your XSIAM capacity.
- **JSON formatting**: Events are serialized into JSON format for XSIAM ingestion.
  - The `Content-Type: application/json` header is included in all requests.
  - The `data` field is represented as a native JSON object, ensuring direct parsing of its contents. This native JSON structure allows XSIAM to efficiently query and analyze the data within the field, for example:

```
{"data": {"resource": "/done", "path": "/done", "isBase64Encoded": false}, "coll
```

Copy

Prerequisite

In XSIAM, create a data collector instance and obtain an authorization token and endpoint URL. See the appropriate Cortex XSIAM documentation below for instructions.

- Cortex XSIAM 2.x
- Cortex XSIAM Premium
- Cortex XSIAM Enterprise
- Cortex XSIAM NG SIEM

Configure an XSIAM Destination

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
    - **Description**: Optionally, enter a description.
    - **XSIAM endpoint**: XSIAM endpoint URL to send events to, following this format `https://api-{tenant external URL}/logs/v1/event`. Defaults to `http://localhost:8088/logs/v1/event`.
3. Under **Authentication**, select an **Authentication method** from the dropdown:
    - **Bearer Token**: Displays an **Auth token** field for you to enter your XSIAM authorization token.
    - **Secret**: Displays an **Auth token (text secret)** drop-down to select a stored secret. A **Create** link is available to store a new, reusable secret.
4. Next, you can configure the following **Optional Settings**:
    - **Backpressure behavior**: Select whether to block, drop, or queue events when all receivers are exerting backpressure. (Causes might include a broken or denied connection, or a rate limiter.) Defaults to `Block`.
    - **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
5. If you assigned the **Backpressure behavior** to `Persistent Queue` you can adjust the Persistent Queue settings.
6. Optionally, you can adjust the Retries, and Advanced settings outlined in the sections below.

7. Select **Save**.
8. Map data to the XSIAM Destination.
9. **Commit & Deploy** your changes.

Map data to the XSIAM Destination

In Cribl Edge, add the **Palo Alto XSIAM** Pack to your Fleet. Navigate to **Processing** > **Packs** from the Fleet submenu, and add the Pack. Once added, you'll be able to assign the Pack to your Route or QuickConnect that maps data from Sources to the XSIAM Destination.

- The Pack includes several Pipelines designed for specific data types.

  Configure the Pack **Route** to use the appropriate **Pipeline** for your data type.

- If you're sending different data types for your XSIAM Destination, create a separate Pack with the appropriate Pipeline for each data type.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.



If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5 GB`. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10

seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.



Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Retries

Honor Retry-After header: Toggle on to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream/Edge limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. Any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. Toggle off to ignore all `Retry-After` headers.

Default Retry Behavior: By default, Cribl Stream/Edge applies predefined retry strategies for certain failed requests to the Destination using an exponential backoff algorithm. You can override these settings by manually configuring the retry behavior for a particular error code.

To retry requests, Cribl Stream/Edge applies the following rules by default:

Status Code	Action
Any in the `1xx`, `3xx` series	Drop the request
`408`, `429` errors	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached
Any other `4xx` series	Drop the request
Any `5xx` series	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached

By default, any `429 (Too Many Requests)` response that is not listed in the retry settings is treated as a non-retryable `4xx` error and dropped. If you want `429` responses retried, ensure `429` is included in the **Settings for failed HTTP requests**.

Settings for failed HTTP requests: You can also manually configure Cribl Stream/Edge to automatically retry requests that receive particular HTTP response status codes.

Upon receiving a response code that's on the list, Cribl Stream/Edge first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream/Edge continues retrying the request without increasing the time between retries any further.

If the sender (which manages the connection to the Destination) is at capacity, it will not accept any incoming events. These incoming events originate internally from a previous stage of the data flow when Destinations send outbound requests to their respective external services, and they include retry requests and new requests. Any events that were already in transit when the sender reached capacity will continue to be processed downstream.

Sender capacity is freed up when an outgoing request succeeds or encounters a non-retryable error. When the sender has available capacity again, it will resume accepting incoming events. This capacity management is influenced by the number of active connections and configured limits, such as concurrency and buffer sizes. If a Pipeline sends events faster than the Destination can process, the buffers may fill up, leading to backpressure and `Sender at capacity` warnings. This backpressure prevents the sender from accepting additional requests until capacity is restored.

For each response code you want to add to the list, select **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: Toggle on to automatically retry requests that have timed out and display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Reject certificates that are not authorized by a trusted CA (for example, the system's CA). Defaults to toggled on.

Round-robin DNS: Toggle on to enable round-robin DNS lookup across multiple IP addresses, IPv4 and IPv6. When a DNS server resolves a Fully Qualified Domain Name (FQDN) to multiple IP addresses, Cribl Edge will sequentially use each address in the order they are returned by the DNS server for subsequent connection attempts.

Compress: Toggle on (default) to compress the payload body before sending.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests per Worker Process. When Cribl Edge hits this limit, it begins throttling traffic to the downstream service. Defaults to `5`. Minimum: `1`. Maximum: `32`.

Events-per-request limit: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values can cause the payload size to be smaller than the configured **Body size limit**. Defaults to `1`.

- Retries happen on this flush interval.
- Any HTTP response code in the `2xx` range is considered success.
- Any response code in the `5xx` range is considered a retryable error, which will not trigger Persistent Queue (PQ) usage.
- Any other response code will trigger PQ (if PQ is configured as the **Backpressure behavior**).

Throttle request rate limit: Maximum number of HTTP requests sent per second. Defaults to `400`, maximum `2000`. Requests exceeding this limit are stored in the persistent queue until capacity becomes available.

Extra HTTP headers: Click **Add Header** to add **Name/Value** pairs to pass as additional HTTP headers. Values will be sent encrypted.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Notes on HTTP-Based Outputs

- To proxy outbound HTTP/S requests, see System Proxy Configuration.
- Cribl Edge will attempt to use keepalives to reuse a connection for multiple requests. After two minutes of the first use, the connection will be thrown away, and a new connection will be reattempted. This is to prevent sticking to a particular Destination when there is a constant flow of events.
- If the server does not support keepalives - or if the server closes a pooled connection while idle - a new connection will be established for the next request.

Troubleshooting

This table outlines common API call errors, showing response codes and log messages for troubleshooting.

Scenario	Response Code	Response Body
Unauthorized access	`401`	N/A (Typically no body for a `401` response)
No Source-Identifier header or an invalid UUID provided in the Source-Identifier header	`400`	`missing or unknown UUID was signaled in Source-Identifier header`
The "Other" (non-XSIAM) integration UUID (`af01292940d7426594d3d3e55ae17ee0`) was signaled in the Source-Identifier but the	`400`	`must signal both vendor and product for non-xsiam integrations`

Scenario	Response Code	Response Body
Vendor and/or Product headers are missing or empty.		
Missing Format header	`400`	`missing format header`
Missing Integration-Identifier header	`400`	`missing Integration-Identifier header`

Configuration Modal

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

12.3.16. CrowdStrike Falcon Next-Gen SIEM Destination

The **CrowdStrike Falcon Next-Gen SIEM** Destination sends data to CrowdStrike Falcon Next-Gen SIEM.



Type: Streaming | TLS Support: Configurable | PQ Support: Yes

Prerequisites

Connector

Within CrowdStrike Falcon Next-Gen SIEM's **Data onboarding** section, create a `Cribl Data Connector`. Select `Cribl` as the connector type during setup to leverage future functionalities and enhancements unavailable through generic HTTP Event Collector (HEC) connectors. This also allows you to filter ingested `Data Sources` by `Cribl Data Connector` for streamlined management of Cribl Edge data streams.

SIEM Parser

Each Next-Gen SIEM connector requires an assigned parser. Map each connector to the appropriate parser for its vendor/product combination. If you have multiple connectors for the same vendor/partner, you can reuse the same parser. If no suitable parser exists, create a new one or clone and modify an existing parser with a similar data format.

By default, CrowdStrike Falcon Next-Gen SIEM parsers expect data in the vendor's original format. Ensure you only send the `_raw` field. For standard data ingestion, use the `passthru` parser. This will preserve the original event format sent to Next-Gen SIEM. However, some products, like Zscaler, usually require specific output formats.

See how to test your SIEM parser.

Syslog

When using a Cribl Edge Syslog Source, explicitly enter `_raw` in the **Fields to keep** list under the Syslog Source's **Optional Settings**. This prevents Cribl Edge from automatically parsing the syslog event into multiple fields, which would unnecessarily increase the event size and ensures that the original syslog event is routed from the Source to the CrowdStrike Falcon Next-Gen SIEM Destination.

Configuring a CrowdStrike Falcon Next-Gen SIEM Destination

1. From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:
    - To configure via QuickConnect, click **Routing** then **QuickConnect** (Cribl Stream) or **Collect** (Cribl Edge). Next, click **Add Destination** and from the resulting drawer's tiles, select **CrowdStrike Falcon Next-Gen SIEM**. Next, click either **Add Destination** or (if displayed) **Select Existing**.
    - To configure via the Routes, click **Data** then **Destinations** (Cribl Stream) or **More** then **Destinations** (Cribl Edge). Select **CrowdStrike Falcon Next-Gen SIEM** from the list of tiles or the **Destinations** list. Next, click **Add Destination** to open a **New Destination** modal.
2. In the Destination modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this Destination definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
    - **Next-Gen SIEM endpoint**: The URL for sending data to CrowdStrike's Falcon Next-Gen SIEM. This URL is generated within the CrowdStrike platform and directs your data to the appropriate SIEM instance for processing.
    - **Request format**: Select `JSON` as the request format. This field defaults to `JSON`, which is the format CrowdStrike Falcon Next-Gen SIEM expects for proper data delivery.



The `Raw` option sends only the event's `_raw` value. This option is intended for advanced use cases where specific data manipulation or custom formatting is required. Use with caution, as it may not be compatible with standard Next-Gen SIEM ingest requirements.

3. Under **Authentication** select an **Authentication method** from the drop-down:
    - **Manual**: Displays a **Next-Gen SIEM authentication token** field for you to enter your authentication token.
    - **Secret**: Displays a **Next-Gen SIEM authentication token (text secret)** drop-down, in which you can select a stored secret that references the API token described above. A **Create** link is available to store a new, reusable secret.
4. Next, you can configure the following **Optional Settings** that you'll find across many Cribl Destinations:
    - **Backpressure behavior**: Select whether to block, drop, or queue events when all receivers are exerting backpressure. (Causes might include a broken or denied connection, or a rate limiter.) Defaults to `Block`.
    - **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
5. Under Processing > **Post-Processing** keep the `cribl_pipe` field to assist with mapping and troubleshooting within NG SIEM. The payload sent to NG SIEM must contain only the `_raw` field. Do not include `_time`, `host`, or any other fields. NG SIEM ingest licensing is based on the size of the entire payload, and default NG SIEM parsers are designed to process only the `_raw` field.

- To retain only the `_raw` field and discard all others, configure an Eval Function within a post-processing Pipeline as follows:
  - In the **Keep fields** input, enter `_raw`.
  - In the **Remove fields** input, enter `*`.

6. Optionally, configure any Persistent Queue, Retries, and Advanced settings outlined in the below sections.
7. Select **Save**, then **Commit & Deploy**.
8. Use the **Test** tab in the Destination's configuration modal to validate the configuration and connectivity of your Destination.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

> On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.
>
> This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.

Mode	Description
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.



If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5` GB. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.



Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Retries

Honor Retry-After header: Toggle on to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream/Edge limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. Any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. Toggle off to ignore all `Retry-After` headers.

Default Retry Behavior: By default, Cribl Stream/Edge applies predefined retry strategies for certain failed requests to the Destination using an exponential backoff algorithm. You can override these settings by manually configuring the retry behavior for a particular error code.



Destinations created before the 4.16.0 release do not support the default `4xx` series retry behavior. If your Destination was created prior to this release and you'd like it to support the behavior, reconfigure

the Destination.

To retry requests, Cribl Stream/Edge applies the following rules by default:

Status Code	Action
Any in the `1xx`, `3xx` series	Drop the request
`401`, `403`, `408`, `429` errors	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached
Any other `4xx` series	Drop the request
Any in the `5xx` series	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached

By default, any `429 (Too Many Requests)` response that is not listed in the retry settings is treated as a non-retryable `4xx` error and dropped. If you want `429` responses retried, ensure `429` is included in the **Settings for failed HTTP requests**.

Settings for failed HTTP requests: You can also manually configure Cribl Stream/Edge to automatically retry requests that receive particular HTTP response status codes.

Upon receiving a response code that's on the list, Cribl Stream/Edge first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream/Edge continues retrying the request without increasing the time between retries any further.

If the sender (which manages the connection to the Destination) is at capacity, it will not accept any incoming events. These incoming events originate internally from a previous stage of the data flow when Destinations send outbound requests to their respective external services, and they include retry requests and new requests. Any events that were already in transit when the sender reached capacity will continue to be processed downstream.

Sender capacity is freed up when an outgoing request succeeds or encounters a non-retryable error. When the sender has available capacity again, it will resume accepting incoming events. This capacity management is influenced by the number of active connections and configured limits, such as concurrency and buffer sizes. If a Pipeline sends events faster than the Destination can process, the buffers may fill up, leading to backpressure and `Sender at capacity` warnings. This backpressure prevents the sender from accepting additional requests until capacity is restored.

For each response code you want to add to the list, select **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: Toggle on to automatically retry requests that have timed out and display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Defaults to toggled on to reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (for example, the system's CA).

Round-robin DNS: Defaults to toggled on to enable round-robin DNS lookup across multiple IP addresses, IPv4 and IPv6. When a DNS server resolves a Fully Qualified Domain Name (FQDN) to multiple IP addresses, Cribl Edge will sequentially use each address in the order they are returned by the DNS server for subsequent connection attempts.

Compress: Defaults to toggled on to compress the payload body before sending.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests per Worker Process. When Cribl Edge hits this limit, it begins throttling traffic to the downstream service. Defaults to `5`. Minimum: `1`. Maximum: `32`. Each request can potentially hit a different HEC receiver.

Body size limit (KB): Maximum size, in KB, of the request body. Defaults to `4096`. Lowering the size can potentially result in more parallel requests and also cause outbound requests to be made sooner.

Events-per-request limit: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values can cause the payload size to be smaller than the configured **Body size limit**. Defaults to `1`.

- Retries happen on this flush interval.
- Any HTTP response code in the `2xx` range is considered success.
- Any response code in the `5xx` range is considered a retryable error, which will not trigger Persistent Queue (PQ) usage.
- Any other response code will trigger PQ (if PQ is configured as the Backpressure behavior).

Extra HTTP headers: Click **Add Header** to add **Name/Value** pairs to pass as additional HTTP headers. Values will be sent encrypted.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Cribl Edge will redact all headers, except non-sensitive headers that you declare below in **Safe headers**.

Safe headers: Add headers that you want to declare as safe to log in plaintext. (Sensitive headers such as `authorization` will always be redacted, even if listed here.) Use a tab or hard return to separate header names.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Notes on HTTP-Based Outputs

- To proxy outbound HTTP/S requests, see System Proxy Configuration.
- Cribl Edge will attempt to use keepalives to reuse a connection for multiple requests. After two minutes of the first use, the connection will be thrown away, and a new connection will be reattempted. This is to prevent sticking to a particular Destination when there is a constant flow of events.
- If the server does not support keepalives - or if the server closes a pooled connection while idle - a new connection will be established for the next request.
- Cribl Edge will pick the first IP in the list for use in the next connection. Enable Round-robin DNS to better balance distribution of events between CrowdStrike Falcon LogScale servers.

Testing Next-Gen SIEM Parser

For data sources like syslog or flat-file logs, you can test your NG SIEM parser using sample data from Cribl Edge. This process is less applicable to API-driven data, as NG SIEM often performs pre-processing on API calls before parsing.

1. **Extract sample `_raw` data**: In Cribl Edge, while editing the relevant Pipeline, open the sample data output and copy the content of the `_raw` field.
2. **Add test event to parser**: In the NG SIEM parser editor, select **Add test** and paste the copied `_raw` data.
3. **Validate parsing**: Select **Test parser** to confirm that the event fields are parsed correctly.



If you are testing a default NG SIEM parser, you must first create a copy of the parser. Then, edit the copy and add the sample test event in the **Test** section.

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

12.3.17. Databricks Destination

The Databricks Destination sends data to Databricks Unity Catalog volumes.



Type: Non-Streaming | TLS Support: N/A | PQ Support: N/A

Configure a Databricks Destination

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this Filesystem definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
    - **Description**: Optionally, enter a description.
    - **Data format**: The output data format defaults to `JSON`. `Raw` and `Parquet` are also available. Selecting `Parquet` (supported only on Linux, not Windows) exposes a **Parquet Settings** left tab, where you **must** configure certain options in order to export data in Parquet format.
3. Next, you can configure the following **Optional Settings**:
    - **Upload path**: Specify an optional directory prefix for files uploaded to the Databricks events volume. It accepts a JavaScript expression (enclosed in quotes or backticks) that evaluates at initialization, allowing dynamic path generation using global variables. For example: `myEventsVolumePath-${C.vars.myVar}`.
      If unset, files are uploaded to the root of the events volume. Use this setting to organize output data by customer, date, or other runtime variables for efficient partitioning and querying in Databricks.
    - **Staging location**: The local filesystem directory where files are temporarily buffered before compression and upload to Databricks. This should be a performant and reliable storage path to avoid bottlenecks or data loss during output operations. Defaults to `$CRIBL_HOME/state/outputs/staging`.

- **Partitioning expression**: JavaScript expression that defines how files are partitioned into subdirectories within the Databricks **Upload path**. Evaluated per event, typically for time-based or custom partitioning. Defaults to `C.Time.strftime(_time ? _time : Date.now()/1000, '%Y/%m/%d')`. This default partitions files by event time (or current time if `_time` is absent), creating a directory structure like `2025/10/29/`.
  If blank, Cribl Edge will fall back to the event's `__partition` field value (if present); or otherwise to the root directory of the **Upload path**.
- **Compression**: Data compression format used before moving to the final destination. Defaults to `gzip` (recommended). This setting is not available when **Data format** is set to `Parquet`.
- **File name prefix expression**: The output file name prefix. Must be a JavaScript expression (which can evaluate to a constant), enclosed in quotes or backticks. Defaults to `CriblOut`.
- **File name suffix expression**: The output file name suffix. Must be a JavaScript expression (which can evaluate to a constant), enclosed in quotes or backticks. Defaults to `` `.${C.env["CRIBL_WORKER_ID"]}.${__format}${__compression === "gzip" ? ".gz" : ""}` ``, where `__format` can be `json` or `raw`, and `__compression` can be `none` or `gzip`.

  > To prevent files from being overwritten, Cribl Edge appends a random sequence of six characters to the end of their names. This also applies to prefix and suffix expressions in file names.
  >
  > For example, if you set the **File name prefix expression** to `CriblExec` and set the **File name suffix expression** to `.csv`, the file name might display as `CriblExec-adPRWM.csv` with `adPRWM` appended.

- **Backpressure Behavior**: Select whether to block or drop events when all receivers are exerting backpressure. (Causes might include an accumulation of too many files needing to be closed.) Defaults to `Block`.
- **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

4. The **Unity Catalog OAuth** settings allow Cribl Edge to authenticate to your Databricks Workspace as an application (known as a service principal) without using a personal access token.
   - **Workspace ID**: Databricks Workspace identifier used for OAuth authentication. See how to find your Workspace ID for details.
   - **Workspace host** (optional): Override the default Databricks workspace hostname. If you leave this blank, Cribl Edge uses `cloud.databricks.com` and constructs the workspace URL as `https://${workspaceId}.cloud.databricks.com`. Set this if your workspace runs in a different Databricks domain, such as `cloud.databricks.us`, `cloud.databricks.mil`, or `azuredatabricks.net`. Enter only the domain portion, without protocol or workspace ID. For example, `azuredatabricks.net`.

- **Client ID**: OAuth client ID for Unity Catalog authentication. See how to create a service principal for details.
- **Client Secret**: Select an existing text secret from the drop-down or add a new text secret by selecting **Create**. See how to create and manage secrets for details.
- **OAuth scope**: The level of permission requested. The default `all-apis` scope requests permission to access all Databricks REST APIs that the service principal has been granted privileges. See authorize service principal access to Databricks with OAuth for details.

5. The **Databricks Settings** define the specific location within the Unity Catalog three-level namespace (`catalog.schema.object`) where Cribl Edge will write data.
   - **Catalog**: The top-level container for your data in Unity Catalog. The default value `main` is the catalog that Databricks automatically creates within a new Unity Catalog metastore.
   - **Schema**: The second level of the namespace, also known as a database. A schema contains tables, views, and volumes. Defaults to `external` to indicate that the data is coming from an external source.
   - **Events volume name**: Name of the events volume in Databricks where files are stored. Defaults to `events`.
6. Optionally, you can adjust the Processing, Parquet, and Advanced settings outlined in the sections below.
7. Select **Save**, then **Commit & Deploy**.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Parquet Settings

To write out Parquet files, note that:

- On Linux, you can use the Cribl Edge CLI's `parquet` command to view a Parquet file, its metadata, or its schema.

- Cribl Edge Workers support Parquet only when running on Linux, not on Windows.
- See Parquet Schemas for pointers on how to avoid problems such as data mismatches.

Automatic schema: Toggle on to automatically generate a Parquet schema based on the events of each Parquet file that Cribl Edge writes. When toggled off (the default), exposes the following additional field:

- **Parquet schema**: Select a schema from the drop-down.



If you need to modify a schema or add a new one, follow the instructions in our Parquet Schemas topic. These steps will propagate the freshest schema back to this drop-down.

Parquet version: Determines which data types are supported, and how they are represented. Defaults to `2.6`; `2.4` and `1.0` are also available.

Data page version: Serialization format for data pages. Defaults to `V2`. If your toolchain includes a Parquet reader that does not support `V2`, use `V1`.

Group row limit: The number of rows that every group will contain. The final group can contain a smaller number of rows. Defaults to `10000`.

Page size: Set the target memory size for page segments. Generally, set lower values to improve reading speed, or set higher values to improve compression. Value must be a positive integer smaller than the **Row group size** value, with appropriate units. Defaults to `1 MB`.

Log invalid rows: Toggle on to output up to 20 unique rows that were skipped due to data format mismatch. Log level must be set to `debug` for output to be visible.

Write statistics: Toggle on (default) if you have Parquet tools configured to view statistics - these profile an entire file in terms of minimum/maximum values within data, numbers of nulls, etc.

Write page indexes: Toggle on (default) if your Parquet reader uses statistics from Page Indexes to enable page skipping. One Page Index contains statistics for one data page.

Write page checksum: Toggle on if you have configured Parquet tools to verify data integrity using the checksums of Parquet pages.

Metadata (optional): The metadata of files the Destination writes will include the properties you add here as key-value pairs. For example, one way to tag events as belonging to the OCSF category for security findings would be to set **Key** to `OCSF Event Class` and **Value** to `2001`.

Advanced Settings

File size limit (MB): Maximum uncompressed output file size. Files of this size will be closed and moved to final output location. Defaults to `32`.

File open time limit (sec): Maximum amount of time to write to a file. Files open for longer than this will be closed and moved to final output location. Defaults to `300`.

Idle time limit (sec): Maximum amount of time to keep inactive files open. Files open for longer than this will be closed and moved to final output location. Defaults to `30`.

Open file limit: Maximum number of files to keep open concurrently. When exceeded, the oldest open files will be closed and moved to final output location. Defaults to `100`.



Cribl Edge will close files when **either** of the `File size limit (MB)` or the `File open time limit (sec)` conditions are met.

Compression level: Specifies the level of compression to apply before moving files to the final destination. Higher compression levels reduce file size but increase CPU usage. Choose from `Best Speed`, `Normal`, or `Best Compression`. The default level, `Best Speed`, will prioritize faster compression over smaller file sizes.

Writing high watermark (KB): Sets the high watermark for writing files, in kilobytes. This determines the buffer size for file writes, impacting performance and memory usage. Enter a numeric value representing the buffer size in kilobytes. The default size, `64` will use a 64 KB buffer for file writes.

Header line: If set, this line will be written to the beginning of each output file, followed by a newline character. This can be useful for adding a header row to CSV files.

Disk space protection: Specifies whether to `Block` (default) or `Drop` incoming events when the disk space falls below the globally defined Min free disk space amount.

Add Output ID: When toggled on (default), adds the **Output ID** field's value to the staging location's file path. This ensures that each Destination's logs will write to its own bucket.

Remove empty staging directories: When toggled on (the default), Cribl Edge deletes empty staging directories after moving files. This prevents the proliferation of orphaned empty directories. When enabled, exposes this additional option:

- **Staging cleanup period**: How often (in seconds) to delete empty directories when **Remove staging dirs** is enabled. Defaults to `300` seconds (every 5 minutes). Minimum configurable interval is `10` seconds; maximum is `86400` seconds (every 24 hours).
- **Directory batch size**: Specifies how many directories are scanned per batch during automatic cleanup of empty staging directories. Set between `10` and `10000`, defaults to `1000`. Higher values speed up cleanup but increase memory usage.

Force close on shutdown: Toggle on to force all staged files to close during an orderly shutdown. This triggers immediate upload or finalization of any in-progress files, regardless of **Idle time**, **File open time**, or **File size** limits. Use this option to minimize data loss risk in ephemeral or auto-scaling environments. This could result in a higher number of small files, as files are closed before reaching their configured thresholds. Ensure your

Worker Group or Fleet **Drain timeout** is set high enough to allow all staged data to be finalized before shutdown completes.

Enable dead-lettering: Toggle on to set a maximum number of retries, and to move files to a designated directory when write failures exceed that limit. This prevents data flow blockage and excessive error logging due to undeliverable files. When enabled, exposes two additional fields:

- **Dead-letter location**: Specify the storage location for undeliverable files. Defaults to `$CRIBL_HOME/state/outputs/dead-letter`.
- **Maximum retry limit**: Configure the retry limit for failed file deliveries. This setting defines how many times the system will attempt to move a file to its intended location before it is deemed undeliverable and placed in the dead-letter directory. Defaults to `20`.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

12.3.18. Datadog Destination

Cribl Edge can send log, metric, and trace events to Datadog. Datadog supports metrics of type `gauge`, `counter`, `rate`, and `distribution` via its REST API.

Cribl Edge sends data via three primary Datadog endpoints, which must be accessible from your Cribl environment:

- Logs: `http-intake.logs.{domain}`
- Metrics: `api.{domain}`
- APM Traces: `trace.agent.{domain}`

The `{domain}` placeholder in the endpoints represents the specific Datadog region you're using. By default, this is set to `datadoghq.com` for the US region. You can configure an alternative region with the **Datadog site** under **Optional Settings** below.

Supported Domains:

- US: `datadoghq.com` (default)
- US3: `us3.datadoghq.com`
- US5: `us5.datadoghq.com`
- Europe: `datadoghq.eu`
- US1-FED: `ddog-gov.com`
- AP1: `ap1.datadoghq.com`
- Custom: `custom.datadoghq.com`



Type: Streaming | TLS Support: Yes | PQ Support: Yes

Configure Cribl Edge to Output to Datadog

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.

- To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.

2. In the **New Destination** modal, configure the following under **General Settings**:
   - **Output ID**: Enter a unique name to identify this Destination definition.
   - **Description**: Optionally, enter a description.
3. Under **Authentication**, select an **Authentication method** from the dropdown:
   - **Manual**: Displays a field for you to enter an **API key** that is available in your Datadog profile.
   - **Secret**: This option exposes an **API key (text secret)** drop-down, in which you can select a stored secret that references the API access token described above. A **Create** link is available to store a new, reusable secret.
   - **API key**: Enter your Datadog organization's API key.
4. Next, you can configure the following **Optional Settings**:
   - **Datadog site**: Select the Datadog region you are sending to. Defaults to `US`; the other options are `US3`, `US5`, `Europe`, `US1-FED`, `AP1`, and `Custom`. Select `Custom` to manually enter your Datadog region's URL if it's not listed.
   - **Send logs as**: Specify the content type to use when sending logs. Defaults to `application/json`, where each log message is represented by a JSON object. The alternative `text/plain` option sends one message per line, with newline `\n` delimiters.
   - **Message field**: Name of the event field that contains the message to send. If not specified, Cribl Edge sends a JSON representation of the whole event (regardless of whether **Send logs as** is set to JSON or plain text).
   - **Source**: Name of the source to send with logs. If you're sending logs as JSON objects (that is, you've selected **Send logs as**: `application/json`), the event's `source` field (if set) will override this value.
   - **Host**: Name of the host to send with logs. If you're sending logs as JSON objects, the event's `host` field (if set) will override this value.
   - **Service**: Name of the service to send with logs. If you're sending logs as JSON objects, the event's `__service` field (if set) will override this value.
   - **Datadog tags**: List of tags to send with logs, such as `env:prod`, `env_staging:east`. These tags enhance search and filtering in Datadog. For log events, these tags are used in the default batching behavior controlled by the **Batch by tags** advanced setting. High tag cardinality can affect performance.



When events also include the `ddtags` internal field, its value must be a single string containing a comma-separated list of key:value pairs (for example, `env:dev,app:myapp`), not an array. Cribl

Edge merges these UI tags with the event `ddtags` value; event tags do not override tags configured here.

- **Severity**: Default value for message severity. If you're sending logs as JSON objects, the event's `__severity` field (if set) will override this value.

Datadog uses the above five fields (`source`, `host`, `__service`, `tags`, and `__severity`) to enhance searches.

- **Allow API key from events**: If toggled on, any API key in the `__agent_api_key` internal field will override the **API key** field's value. This option is useful if events originate from multiple Datadog Agent Sources, each configured with a different API key. (For further details, see Managing API Keys.)
- **Backpressure behavior**: Specify whether to block, drop, or queue events when all receivers are exerting backpressure. Defaults to `Block`.
- **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

5. Optionally, you can adjust the Persistent Queue, Processing, Retries, and Advanced settings outlined in the sections below.
6. Select **Save**, then **Commit & Deploy**.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

> This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

> If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5 GB`. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.



Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Retries

Honor Retry-After header: Toggle on to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream/Edge limits the delay to 180 seconds even if the `Retry-After`

header specifies a longer delay. Any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. Toggle off to ignore all `Retry-After` headers.

Default Retry Behavior: By default, Cribl Stream/Edge applies predefined retry strategies for certain failed requests to the Destination using an exponential backoff algorithm. You can override these settings by manually configuring the retry behavior for a particular error code.

To retry requests, Cribl Stream/Edge applies the following rules by default:

Status Code	Action
Any in the `1xx, 3xx` series	Drop the request
`408, 429` errors	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached
Any other `4xx` series	Drop the request
Any `5xx` series	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached

By default, any `429 (Too Many Requests)` response that is not listed in the retry settings is treated as a non-retryable `4xx` error and dropped. If you want `429` responses retried, ensure `429` is included in the **Settings for failed HTTP requests**.

Settings for failed HTTP requests: You can also manually configure Cribl Stream/Edge to automatically retry requests that receive particular HTTP response status codes.

Upon receiving a response code that's on the list, Cribl Stream/Edge first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream/Edge continues retrying the request without increasing the time between retries any further.

If the sender (which manages the connection to the Destination) is at capacity, it will not accept any incoming events. These incoming events originate internally from a previous stage of the data flow when Destinations send outbound requests to their respective external services, and they include retry requests and new requests. Any events that were already in transit when the sender reached capacity will continue to be processed downstream.

Sender capacity is freed up when an outgoing request succeeds or encounters a non-retryable error. When the sender has available capacity again, it will resume accepting incoming events. This capacity management is influenced by the number of active connections and configured limits, such as concurrency and buffer sizes. If a Pipeline sends events faster than the Destination can process, the buffers may fill up, leading to backpressure

and `Sender at capacity` warnings. This backpressure prevents the sender from accepting additional requests until capacity is restored.

For each response code you want to add to the list, select **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: Toggle on to automatically retry requests that have timed out and display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Toggle on to reject certificates that are **not** authorized by a CA in the **CA certificate path**, nor by another trusted CA (for example, the system's CA).

Round-robin DNS: Toggle on to enable round-robin DNS lookup across multiple IP addresses, IPv4 and IPv6. When a DNS server resolves a Fully Qualified Domain Name (FQDN) to multiple IP addresses, Cribl Edge will sequentially use each address in the order they are returned by the DNS server for subsequent connection attempts.

Compress: Toggle on (default; recommended) to compress the payload body before sending.

Send counter metrics as 'count': Unless toggled on, Datadog might transform `counter` metrics to `gauge`. Learn more about Datadog metrics types.

Batch by tags: When toggled on (default), log events are batched using both the API key and **Datadog tags** to optimize logical grouping. However, high cardinality in Datadog tags (many unique tag values) can result in

excessive concurrent requests and potential `408` errors from Datadog. Toggling this setting off changes the batching behavior to group log events solely by API key, reducing batch count and improving performance.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests per Worker Process. When Cribl Edge hits this limit, it begins throttling traffic to the downstream service. Defaults to `5`. Minimum: `1`. Maximum: `32`.

Body size limit (KB): Maximum size of the request body before compression. Defaults to `4096` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Body size limit** value until downstream receivers reliably accept all events.

Buffer memory limit (KB): Total amount of memory used to buffer outgoing requests waiting to be sent. If left blank, defaults to 5 times the max body size (if set). If 0, no limit is enforced. This provides granular control over the memory allocated for buffering outgoing batched requests. Increasing the limit allows batches to grow larger before being flushed, improving efficiency for data with high cardinality (a large number of unique batches). Finding the optimal balance between efficient data transfer and memory usage involves adjusting both the **Buffer memory limit** and **Body size limit** settings.

Events-per-request limit: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (s): Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Extra HTTP headers: Name-value pairs to pass as additional HTTP headers. Values will be sent encrypted.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Cribl Edge will redact all headers, except non-sensitive headers that you declare below in **Safe headers**.

Safe headers: Add headers to declare them as safe to log in plaintext. (Sensitive headers such as `authorization` will always be redacted, even if listed here.) Use a tab or hard return to separate header names.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Internal Fields

Cribl Edge uses a set of internal fields to assist in forwarding data to a Destination.

If an event contains the internal field `__criblMetrics`, Cribl Edge will send it to Datadog as a metric event. Events that do not contain `__criblMetrics` are sent as logs or traces, based on their structure. No internal

fields are supported for metric events or trace events.

You can use these fields to **override** outbound event values for log events:

- `ddtags`: Optional tags to send with log events. The value must be a single string containing a comma-separated list of key:value pairs, not an array. For example: `env:dev,app:myapp`. If you send `ddtags` as an array (such as `["env:dev","app:myapp"]`), Datadog rejects the payload. When both the Datadog Destination Datadog tags setting and the event `ddtags` field are set, Cribl Edge merges them into a single tag list before sending them to Datadog.
- `__service`
- `__severity`

Proxying Requests

If you need to proxy HTTP/S requests, see System Proxy Configuration.

For More Information

You might find these Datadog references helpful:

- Submit Metrics
- Send Logs
- Metrics Types

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

12.3.19. Dynatrace HTTP Destination

The Dynatrace HTTP Destination sends logs to Dynatrace's SaaS or ActiveGate endpoints.



Type: Streaming | TLS Support: Configurable | PQ Support: Yes

Configure a Dynatrace HTTP Destination

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing**.
    - To configure via the Routes, select **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the Destination modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this Destination definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**. A **Description** is optional.
    - **Endpoint**: Select the type of Dynatrace endpoint where you want to send your data.
        - **Cloud**: Sends data to Dynatrace's SaaS endpoint. Provide your Dynatrace **Environment ID**. This ID uniquely identifies your Dynatrace environment and is required for routing data correctly. You can find your environment ID in your Dynatrace account settings.
        - **ActiveGate**: Choose this option if you are routing data through an ActiveGate. Provide your Dynatrace **Environment ID** and **ActiveGate domain** in the additional fields that appear. Your ActiveGate domain is displayed in your Dynatrace settings in the **Details** section of an ActiveGate under the **Hostname** field. The format should be a fully qualified domain name (FQDN) or IP address. For example, `https://{activeGate-domain}:9999/e/{environment-id}/api/v2/logs/ingest`.
        - **Manual**: Select this option if you want to manually specify the endpoint URL. This is useful for custom setups or non-standard configurations. You will need to enter the full **URL** in the field that appears. Can be overwritten by an event's `__url` field.
3. Next, you can configure the following **Optional Settings**:
    - **Backpressure behavior**: Select whether to block, drop, or queue events when all receivers are exerting backpressure. (Causes might include a broken or denied connection, or a rate limiter.)

Defaults to `Block`.

- **Tags**: Optionally, add tags that you can use to filter and group Destinations in Cribl Edge's **Manage Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

4. Select **Authentication** from the sidebar and select an **Authentication type**.
    - `Auth token`: Provide a Dynatrace API access token to authenticate in the **Token** field. This is the most common method for securing data transmission to Dynatrace. You can generate this token in your Dynatrace account settings under the API tokens section. Ensure that the token has the necessary permissions for the data you are sending.
    - `Token (text secret)`: Create or select a Cribl stored text secret that contains your Dynatrace API access token.
5. Optionally, configure any Persistent Queue, Processing, Retries, and Advanced settings outlined in the below sections.
    - **Persistent Queue** ensures data durability and reliability by buffering and preserving incoming events when the receiver is down or exhibiting backpressure.
    - **Processing Settings** allow you to define pre-processing and post-processing Pipelines. This includes adding metadata, applying transformations, and conditioning the data before it's sent.
    - **Retries** allow you to specify how the system should handle different HTTP response status codes and timeouts, ensuring reliable data delivery.
    - **Advanced Settings** offer additional configuration options that enhance the flexibility and performance of data transmission.



To track data sent via Cribl, you can add a custom attribute in the **Extra HTTP headers** section under Advanced Settings. For example, select `Add Header` and enter `cribl` as the name and `yes` as the value.

6. Select **Save**, then **Commit & Deploy**.
7. Use the **Test** tab in the Destination's configuration modal to validate the configuration and connectivity of your Destination.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues

- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5 GB`. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.



Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.

- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Retries

Honor Retry-After header: Toggle on to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream/Edge limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. Any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. Toggle off to ignore all `Retry-After` headers.

Default Retry Behavior: By default, Cribl Stream/Edge applies predefined retry strategies for certain failed requests to the Destination using an exponential backoff algorithm. You can override these settings by manually configuring the retry behavior for a particular error code.

To retry requests, Cribl Stream/Edge applies the following rules by default:

Status Code	Action
Any in the `1xx, 3xx` series	Drop the request
`408, 429` errors	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached
Any other `4xx` series	Drop the request
Any `5xx` series	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached

By default, any `429 (Too Many Requests)` response that is not listed in the retry settings is treated as a non-retryable `4xx` error and dropped. If you want `429` responses retried, ensure `429` is included in the **Settings for failed HTTP requests**.

Settings for failed HTTP requests: You can also manually configure Cribl Stream/Edge to automatically retry requests that receive particular HTTP response status codes.

Upon receiving a response code that's on the list, Cribl Stream/Edge first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream/Edge continues retrying the request without increasing the time between retries any further.

If the sender (which manages the connection to the Destination) is at capacity, it will not accept any incoming events. These incoming events originate internally from a previous stage of the data flow when Destinations send outbound requests to their respective external services, and they include retry requests and new

requests. Any events that were already in transit when the sender reached capacity will continue to be processed downstream.

Sender capacity is freed up when an outgoing request succeeds or encounters a non-retryable error. When the sender has available capacity again, it will resume accepting incoming events. This capacity management is influenced by the number of active connections and configured limits, such as concurrency and buffer sizes. If a Pipeline sends events faster than the Destination can process, the buffers may fill up, leading to backpressure and `Sender at capacity` warnings. This backpressure prevents the sender from accepting additional requests until capacity is restored.

For each response code you want to add to the list, select **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: Toggle on to automatically retry requests that have timed out and display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Reject certificates that are **not** authorized by a CA in the **CA certificate path**, nor by another trusted CA (for example, the system's CA). Defaults to toggled on.

Round-robin DNS: Toggle on to enable round-robin DNS lookup across multiple IP addresses, IPv4 and IPv6. When a DNS server resolves a Fully Qualified Domain Name (FQDN) to multiple IP addresses, Cribl Edge will sequentially use each address in the order they are returned by the DNS server for subsequent connection attempts.

Compress: Toggle on (default) to compress the payload body before sending.

Keep alive: By default, Cribl Edge sends `Keep-Alive` headers to the remote server and preserves the connection from the client side up to a maximum of 120 seconds. Toggle this off if you want Cribl Edge to close the connection immediately after sending a request.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests per Worker Process. When Cribl Edge hits this limit, it begins throttling traffic to the downstream service. Defaults to `5`. Minimum: `1`. Maximum: `32`.

Body size limit (KB): Maximum size of the request body before compression. Defaults to `4096` KB. You can set this limit to as high as 5 MB (`5000` KB). Be aware that high values can cause high memory usage per Worker Node, especially if a dynamically constructed URL (see Internal Fields) causes this Destination to send events to more than one URL. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Body size limit** value until downstream receivers reliably accept all events.

Buffer memory limit (KB): Total amount of memory used to buffer outgoing requests waiting to be sent. If left blank, defaults to 5 times the max body size (if set). If 0, no limit is enforced. This provides granular control over the memory allocated for buffering outgoing batched requests. Increasing the limit allows batches to grow larger before being flushed, improving efficiency for data with high cardinality (a large number of unique batches). Finding the optimal balance between efficient data transfer and memory usage involves adjusting both the **Buffer memory limit** and **Body size limit** settings.

Events-per-request limit: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Extra HTTP headers: Name-value pairs to pass to all events as additional HTTP headers. Values will be sent encrypted. You can also add headers dynamically on a per-event basis in the `__headers` field. See Internal Fields below.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Cribl Edge will redact all headers, except non-sensitive headers that you declare below in **Safe headers**.

Safe headers: Add headers to declare them as safe to log in plaintext. (Sensitive headers such as `authorization` will always be redacted, even if listed here.) Use a tab or hard return to separate header names.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Internal Fields

Cribl Edge uses a set of internal fields to assist in forwarding data to a Destination.

Fields for this Destination:

- `__criblMetrics`
- `__url`
- `__headers`

If an event contains the internal field `__url`, that field's value will override the **General Settings** > **Endpoint** value. This way, you can determine the endpoint URL dynamically.

For example, you could create a Pipeline containing an Eval Function that adds the `__url` field, and connect that Pipeline to your Dynatrace HTTP Destination. Configure the Eval Function to set `__url`'s value to a URL that varies depending on a global variable, or on some property of the event, or on some other dynamically-generated value that meets your needs.

If an event contains the internal field `__headers`, that field's value will be a JSON object containing Name-value pairs, each of which defines a header. By creating Pipelines that set the value of `__headers` according to conditions that you specify, you can add headers dynamically.

For example, you could create a Pipeline containing Eval Functions that add the `__headers` field, and connect that Pipeline to your Dynatrace HTTP Destination. Configure the Eval Functions to set `__headers` values to Name-value pairs that vary depending on some properties of the event, or on dynamically-generated values that meet your needs.

Here's an overview of how to add headers:

- To add "dynamic" (a.k.a. "custom") headers to some events but not others, use the `__headers` field.
- To define headers to add to **all** events, use **Advanced Settings** > **Extra HTTP Headers**.
- An event can include headers added by both methods. So if you use `__headers` to add `{ "api-key": "foo" }` and **Extra HTTP Headers** to add `{ "goat": "Kid A" }`, you'll get both headers.
- Headers added via the `__headers` field take precedence. So if you use `__headers` to add `{ "api-key": "foo" }` and **Extra HTTP Headers** to add `{ "api-key": "bar" }`, you'll get only one header, and that will be `{ "api-key": "foo" }`.

Notes on HTTP-Based Outputs

- To proxy outbound HTTP/S requests, see System Proxy Configuration.
- Cribl Edge will attempt to use keepalives to reuse a connection for multiple requests. After two minutes of the first use, it will throw away the connection and attempt a new one. This is to prevent sticking to a particular destination when there is a constant flow of events.
- If the server does not support keepalives (or if the server closes a pooled connection while idle), Cribl Edge will establish a new connection for the next request.
- When resolving the Destination's hostname, Cribl Edge will pick the first IP in the list for use in the next connection. Enable **Round-robin DNS** to better balance the distribution of events among destination cluster nodes.

Troubleshoot

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

12.3.20. Dynatrace OTLP Destination

The Dynatrace OTLP Destination sends telemetry data, including traces, logs, and metrics, to Dynatrace using the OpenTelemetry Protocol (OTLP). It can send data to various Dynatrace endpoints, including SaaS, Environment ActiveGate, and Containerized Environment ActiveGate.

To get started, connecting to Dynatrace is simple. Select your endpoint and provide your access token. Need more detailed instructions or configuration options? See below!



Type: Streaming | TLS Support: Configurable | PQ Support: Yes

Configure a Dynatrace OTLP Destination

In Cribl Edge, set up a Dynatrace OTLP Destination.

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing**.
    - To configure via the Routes, select **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the Destination modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this output definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**. A **Description** is optional.
    - **OTLP version**: Already set to `1.3.1` and cannot be changed.
    - **Protocol**: Already set to `HTTP` and cannot be changed. Uses Binary Protobuf encoding (does not support JSON Protobuf).
    - **Endpoint type**: Select the type of Dynatrace endpoint where you want to send your data.
        - **SaaS**: Sends data to Dynatrace's SaaS endpoint.
        - **ActiveGate**: Choose this option if you are routing data through an ActiveGate.
    - **Endpoint**: The endpoint URL for the Dynatrace environment. You'll need to provide your Dynatrace environment ID. This ID uniquely identifies your Dynatrace environment and is required for routing data correctly. You can find your environment ID in your Dynatrace account settings.

        - **SaaS** defaults to `https://{your-environment-id}.live.dynatrace.com/api/v2/otlp`. Replace the `{your-environment-id}` placeholder with your environment ID.
        - **ActiveGate** defaults to `https://{your-activegate-domain}:9999/e/{your-environment-id}/api/v2/otlp`. Replace the `{your-activegate-domain}` placeholder with your ActiveGate domain, and `{your-environment-id}` with your environment ID. Your ActiveGate domain is displayed in your Dynatrace settings in the **Details** section of an ActiveGate under the **Hostname** field. The format should be a fully qualified domain name (FQDN) or IP address.
        - You can specify a custom endpoint using either a valid URL or an IP address (IPv4 or IPv6). Enclose IPv6 addresses in square brackets (for example, `[2001:db8:85a3:0000:0000:8a2e:0370:7334]`).
        - To proxy outbound HTTP/S requests, see System Proxy Configuration.
    - **Auth token (text secret)**: Create or select a Cribl stored text secret that contains your Dynatrace API access token, ensuring secure authentication for data transmission. You can generate this token in your Dynatrace account settings under the API tokens section. Ensure that the token has the necessary permissions for the data you are sending.
    - **Optional Settings**:
        - You can customize the path instead of using the default cloud endpoints. This is useful when integrating with a hosted Lightstep satellite.
        - **Backpressure behavior**: Whether to `Block`, `Drop`, or `Persistent Queue` events when all receivers are exerting backpressure.
        - **Tags**: Optionally, add tags that you can use to filter and group Destinations in Cribl Edge's **Manage Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
3. The **Processing Settings**, **Retries**, and **Advanced Settings** provide granular control over how data is handled and transmitted.
    - Processing Settings allow you to define pre-processing and post-processing Pipelines, ensuring that events conform to the relevant Protobuf specifications for traces, metrics, and logs. This includes adding metadata, applying transformations, and conditioning the data.
    - Retries allow you to specify how the system should handle different HTTP response status codes and timeouts, ensuring reliable data delivery.
    - Advanced Settings offer additional configuration options that enhance the flexibility and performance of data transmission.
4. Select **Save**, then **Commit & Deploy**.
5. Use the **Test** tab in the Destination's configuration modal to validate the configuration and connectivity of your Destination.

Custom HTTP Path

To send data to a custom location, override the default endpoint paths. When the **Protocol** is set to `HTTP`, this Destination sends data to the following endpoints by default:

- Traces: `<endpoint>/v1/traces`
- Metrics: `<endpoint>/v1/metrics`
- Logs: `<endpoint>/v1/logs`

`<endpoint>` refers to the base URL you specified in the **Endpoint** setting above.

To override a default path, provide only the desired path in the corresponding field: **Traces endpoint override**, **Metrics endpoint override**, or **Logs endpoint override**.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

> On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.
>
> This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.

Mode	Description
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5 GB`. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.



Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output - both metric events, as dimensions; and log events, as labels. Supports wildcards.

By default, includes `cribl_pipe` (Cribl Edge Pipeline that processed the event).

Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Retries

Honor Retry-After header: Toggle on to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream/Edge limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. Any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. Toggle off to ignore all `Retry-After` headers.

Default Retry Behavior: By default, Cribl Stream/Edge applies predefined retry strategies for certain failed requests to the Destination using an exponential backoff algorithm. You can override these settings by manually configuring the retry behavior for a particular error code.

Destinations created before the 4.16.0 release do not support the default `4xx` series retry behavior. If your Destination was created prior to this release and you'd like it to support the behavior, reconfigure the Destination.

To retry requests, Cribl Stream/Edge applies the following rules by default:

Status Code	Action
Any in the `1xx, 3xx` series	Drop the request
`401, 403, 408, 429` errors	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached
Any other `4xx` series	Drop the request
Any in the `5xx` series	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached

By default, any `429 (Too Many Requests)` response that is not listed in the retry settings is treated as a non-retryable `4xx` error and dropped. If you want `429` responses retried, ensure `429` is included in the **Settings for failed HTTP requests**.

Settings for failed HTTP requests: You can also manually configure Cribl Stream/Edge to automatically retry requests that receive particular HTTP response status codes.

Upon receiving a response code that's on the list, Cribl Stream/Edge first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream/Edge continues retrying the request without increasing the time between retries any further.

If the sender (which manages the connection to the Destination) is at capacity, it will not accept any incoming events. These incoming events originate internally from a previous stage of the data flow when Destinations send outbound requests to their respective external services, and they include retry requests and new requests. Any events that were already in transit when the sender reached capacity will continue to be processed downstream.

Sender capacity is freed up when an outgoing request succeeds or encounters a non-retryable error. When the sender has available capacity again, it will resume accepting incoming events. This capacity management is influenced by the number of active connections and configured limits, such as concurrency and buffer sizes. If a Pipeline sends events faster than the Destination can process, the buffers may fill up, leading to backpressure and `Sender at capacity` warnings. This backpressure prevents the sender from accepting additional requests until capacity is restored.

For each response code you want to add to the list, select **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: Toggle on to automatically retry requests that have timed out and display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Toggle on to reject certificates that are **not** authorized by a CA in the **CA certificate path**, nor by another trusted CA (e.g., the system's CA).

Round-robin DNS: Toggle on to enable round-robin DNS lookup across multiple IP addresses, IPv4 and IPv6. When a DNS server resolves a Fully Qualified Domain Name (FQDN) to multiple IP addresses, Cribl Edge will sequentially use each address in the order they are returned by the DNS server for subsequent connection attempts.

Keep alive: By default, Cribl Edge sends `Keep-Alive` headers to the remote server and preserves the connection from the client side up to a maximum of 120 seconds. Toggle off if you want Cribl Edge to close the connection immediately after sending a request.

Compression: Compression type to apply to messages sent to the OpenTelemetry endpoint. `Gzip` (the default) and `None` are available for both protocols.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30` sec.

Request concurrency: Maximum number of concurrent requests before blocking. This is set per Worker Process. Defaults to `5`.

Body size limit (KB): Maximum size of the request body before compression. Defaults to `2048` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Body size limit** value until downstream receivers reliably accept all events. See Dynatrace's best practices.

Flush period (sec): Maximum time between requests. Low values could cause the payload size to be smaller than the configured **Max body size**. Defaults to `1` sec.

Extra HTTP headers: Select **Add Header** to define additional HTTP headers to pass to all events. Each row is a **Name**-**Value** pair. Values will be sent encrypted.

Safe headers: Add headers here to declare them as safe to log in plaintext. (Sensitive headers such as `authorization` will always be redacted, even if listed here.) Use a tab or hard return to separate header names.

Environment: Optionally, specify a single Git branch on which to enable this configuration. If this field is empty, the config will be enabled everywhere.

Protocol and Transport Support

This Destination leverages the OpenTelemetry Protocol (OTLP) v1.3.1 to send observability data to Dynatrace. OTLP defines Protocol buffer (Protobuf) schemas for its payloads (requests and responses). With the HTTP transport, this Destination supports Binary Protobuf payload encoding, but currently does not support JSON Protobuf.

When configuring Pipelines (including pre-processing and post-processing Pipelines), you need to ensure that events sent to this Destination conform to the relevant Protobuf specification.

- For traces, opentelemetry/proto/trace/v1/trace.proto
- For metrics, opentelemetry/proto/metrics/v1/metrics.proto
- For logs, opentelemetry/proto/logs/v1/logs.proto

This Destination will drop non-conforming events. If the Destination encounters a parsing error when trying to convert an event to OTLP, it discards the event and Cribl Edge logs the error.

Troubleshoot

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

12.3.21. Exabeam Security Operations Platform Destination

Cribl Edge supports sending data to the Exabeam security operations platform or (as some people think of it) SIEM.

The Exabeam Destination supersedes and improves upon the "old way" to get data from Cribl Edge to Exabeam using the Cribl Webhook Destination.

Type: Non-Streaming | TLS Support: Yes | PQ Support: No

For examples of using Cribl Edge to optimize inbound events for Exabeam, see:

- Splunk to Exabeam, which demonstrates combining several Cribl Functions to transform events before forwarding to Exabeam.
- Several Exabeam Packs on the Cribl Packs Dispensary, which provide processing Pipelines that you can import and adapt to your needs.

Understand How Exabeam and Cribl Edge Work Together

Exabeam parsers expect to receive events in original, unmodified form. To make sure this happens, you'll sometimes need to use a combination of Event Breakers and Pipelines. In general, knowing this can help you avoid problems or troubleshoot any that do crop up.

When you send data to Exabeam, Cribl Packs are your friends, because Cribl and Exabeam have partnered to create a set of Exabeam-specific Packs, each of which ensures that events from one common data source arrives at Exabeam in precisely the expected form. Cribl Packs also give you the option of dropping events on a per-data-source basis, so that Cribl Edge never sends irrelevant events to Exabeam. To view the Exabeam-specific Cribl Packs, navigate to **Processing** > **Packs**. Then select **Add Pack** > **Add from Dispensary**, and enter `Exabeam` in the search field.

Cribl strongly recommends that you use **only** the Exabeam-specific, Cribl-authored Packs with this Destination. Other Packs can perform transformations - especially reduction - that prevent the Exabeam parsers from triggering.

When the original data source is Microsoft Active Directory (AD) or LDAP, you'll need to set up an Exabeam Site Collector to perform the initial data capture.

Because Exabeam runs in the Google Cloud Storage (GCS) platform, some setting names include references to GCS.

Finally, Cribl recommends placing Cribl Edge Edge Nodes as close as possible to the sources of the events you want to send to Exabeam. This reduces the length of the path that data must traverse before Cribl Edge optimizes it with compression, reduction, dropping events, and/or redacting sensitive information.

Configure Cloud Storage Permissions

For Cribl Edge to send data to Exabeam buckets, the following access permissions must be set on the Cloud Storage side:

- Fine-grained access control must be enabled on the buckets.
- The Google service account or user must have the Storage Admin or Owner role.

For details, see the Cloud Storage Overview of Access Control and Understanding Roles documentation.

Create the Cribl Cloud Collector in Exabeam

Log in to the Exabeam Security Operations Platform, and on the **Home** page, select **Cloud Collectors**. In the resulting page, select the **Cribl** tile to see a list of Cribl Collectors (the first time you do this, the list will be empty).

Select **New Collector**, and in the resulting **Collector for Cribl** drawer:

1. Enter a name for your new Cribl Cloud Collector.
2. Optionally, in **Advanced Settings**, configure the **Metadata** value **TIMEZONE**
3. If your desired site already exists, select it from the **SITE** drop-down.
    - If your desired site does not exist yet, skip the drop-down and select the **manage your sites** link underneath. This will take you to a separate UI where you will enter a value for the **Metadata** value **SITENAME**, and Exabeam will generate the associated metadata value **SITE ID**. When you have done this, Exabeam will take you back to the previous UI, and you must then select your newly-created site from the **SITE** drop-down.
4. Select **Install**.

A `Hooray!` confirmation modal will now replace the **Collector for Cribl** drawer. In this modal:

1. Select the copy icon to the right of the encoded **Connection String**. Save this string in a secure and handy location - it is required when configuring the Cribl Edge Exabeam Destination.
2. Select **Go to Overview** to exit the modal.



Although optional, configuring metadata is useful because it allows a single instance of the Exabeam SIEM to differentiate between data from multiple sources. This is relevant for large enterprises whose IT infrastructure comprises many sources whose IP addresses overlap.

Configure Cribl Edge to Output to Cloud Storage Destinations

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this Exabeam Destination definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
    - **Description**: Optionally, enter a description.
    - Select **Autofill with Exabeam Connection String** to automatically enter the values for the following **Exabeam Settings**. Or enter them directly.
    - **Google Cloud Storage bucket**: Name of the destination bucket - in Exabeam's Cribl Cloud Collector this is called **GCS Bucket Name**. This value can be a constant, or a JavaScript expression that can be evaluated only at init time. For example, you can reference a Global Variable like this: `myBucket-${C.vars.myVar}`.
    - **Google Cloud Storage bucket region**: In Exabeam's Cribl Cloud Collector this is called **GCS Bucket Region**. It is the GCS region where the bucket is located.
    - **Collector instance ID**: In Exabeam's Cribl Cloud Collector this is called **Instance ID**.
    - **Access key**: In Exabeam's Cribl Cloud Collector this is called **Access Key**.
    - **Secret**: In Exabeam's Cribl Cloud Collector this is called **Secret Key**.
    - **Site name**: In Exabeam's Cribl Cloud Collector this is called **SITENAME**. This determines where your data is categorized. Set a static site name or dynamically assign it using a JavaScript expression. For example, to dynamically set the site name based on the `cribl_pipeline` field you'd use the expression `${C.vars.cribl_pipeline}`. This assigns the site name based on the Pipeline value.
    - **Site ID**: In Exabeam's Cribl Cloud Collector this is called **SITE ID**.
    - **Timezone offset**: In Exabeam's Cribl Cloud Collector this is called **TIMEZONE**.
3. Optionally, you can adjust the Processing and Advanced settings outlined in the sections below.
4. Select **Save**, then **Commit & Deploy**.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports `c*` wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Advanced Settings

Backpressure behavior: Select whether to block or drop events when all receivers are exerting backpressure. (Causes might include an accumulation of too many files needing to be closed.) Defaults to `Block`.

File open time limit (sec): Maximum amount of time to write to a file. Files open for longer than this limit will be closed and moved to final output location. Defaults to `300` seconds (5 minutes).

Idle time limit (sec): Maximum amount of time to keep inactive files open. Files open for longer than this limit will be closed and moved to final output location. Defaults to `30`.

Open file limit: Maximum number of files to keep open concurrently. When exceeded, the oldest open files will be closed and moved to final output location. Defaults to `100`.



Cribl Edge will also close any files larger than 10 MB.

Disk space protection: Specifies whether to `Block` (default) or `Drop` incoming events when the disk space falls below the globally defined Min free disk space amount.

Remove empty staging directories: When toggled on (the default), Cribl Edge deletes empty staging directories after moving files. This prevents the proliferation of orphaned empty directories. When enabled, exposes this additional option:

- **Staging cleanup period**: How often (in seconds) to delete empty directories when **Remove staging dirs** is enabled. Defaults to `300` seconds (every 5 minutes). Minimum configurable interval is `10` seconds; maximum is `86400` seconds (every 24 hours).

- **Directory batch size**: Specifies how many directories are scanned per batch during automatic cleanup of empty staging directories. Set between `10` and `10000`, defaults to `1000`. Higher values speed up cleanup but increase memory usage.

Enable dead-lettering: Toggle this on to set a maximum number of retries, and to move files to a designated directory when write failures exceed that limit. This prevents data flow blockage and excessive error logging due to undeliverable files. When enabled, exposes two additional fields:

- **Dead-letter location**: Specify the storage location for undeliverable files. Defaults to `$CRIBL_HOME/state/outputs/dead-letter`.
- **Maximum retry limit**: Configure the retry limit for failed file deliveries. This setting defines how many times the system will attempt to move a file to its intended location before it is deemed undeliverable and placed in the dead-letter directory. Defaults to `20`.

Reuse connections: Whether to reuse connections between requests. Toggling on (default) can improve performance.

Reject unauthorized certificates: Whether to accept certificates (such as self-signed certificates, for example) that cannot be verified against a valid Certificate Authority. Defaults to toggled on.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Tags: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Internal Fields

Cribl Edge uses a set of internal fields to assist in forwarding data to a Destination.

Field for this Destination:

- `__partition`

Troubleshooting

Verify that you are sending the original, unmodified events. Study the applicable Exabeam parsers to make sure you understand exactly what they're doing. If you need to modify events, make sure that when you do, the relevant parsers still trigger.

Nonspecific messages from Google Cloud of the form `Error: failed to close file` can indicate problems with the permissions listed above.

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

12.3.22. Microsoft Fabric Real-Time Intelligence

Cribl Edge supports sending data to Eventstreams in Microsoft Fabric Real-Time Intelligence.

Type: Streaming | TLS Support: Configurable | PQ Support: Yes

Before you create a Microsoft Fabric Real-Time Intelligence Destination in Cribl Edge, create a **Cribl** data source in Microsoft Fabric. Microsoft Fabric will then provision the connection values (like the bootstrap server) that you must use when you configure a Microsoft Fabric Real-Time Intelligence Destination in Cribl Edge.

Configuring Cribl Edge to Output to Microsoft Fabric Real-Time Intelligence

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: A unique name to identify this Microsoft Fabric Real-Time Intelligence Destination. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
    - **Description**: Optionally, enter a description.
    - **Bootstrap server**: The value of the **Bootstrap server** setting for the **Cribl** data source in Microsoft Fabric.
    - **Topic name**: The value of the **Topic name** setting for the **Cribl** data source in Microsoft Fabric. Can be overwritten using the `__topicOut` field.
3. Next, you can configure the following **Optional Settings**:
    - **Acknowledgments**: Control the number of required acknowledgments. Defaults to `Leader`. Setting this control to `Leader` or `All` may significantly slow throughput. If the impact is too great for your use case, set this control to `None`.
    - **Record data format**: Format to use to serialize events before writing. Defaults to `JSON`.

        - `JSON` serializes the entire event from your Pipeline, including all metadata and enriched fields, into a single JSON object.
        - `Field _raw` sends only the contents of the `_raw` field as the payload. This option provides maximum throughput and minimal overhead. If using raw data, send the data in CSV or JSON format.
    - **Backpressure behavior**: Whether to block, drop, or queue events when all receivers are exerting backpressure. Defaults to `Block`.
    - **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
4. Optionally, you can adjust the TLS, Authentication, Processing, Retries and Advanced settings outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

TLS Settings (Client Side)

Enabled: Defaults to toggled on.

Validate server certs: Defaults to toggled on.

Authentication

Authentication parameters to use when connecting to Microsoft Fabric. Using TLS is highly recommended.

SASL mechanism: SASL (Simple Authentication and Security Layer) authentication mechanism to use. Defaults to `PLAIN`, which exposes Basic Authentication options from the **SASL mechanism** settings in the **Cribl** data source in Microsoft Fabric. Select `OAUTHBEARER` to enable OAuth Authentication via a different set of options.

Basic Authentication

Selecting the `PLAIN` SASL mechanism provides the options listed in this section.

SASL JASS username: The value is `$ConnectionString` (immutable).

SASL JASS password: The stored secret that references the **SASL JASS password-primary** or **SASL JASS password-secondary** in the **Cribl** data source in Microsoft Fabric. The secret can reside in Cribl Edge's internal secrets manager or (if enabled) in an external KMS. A **Create** link is available if you need a new secret.

OAuth Authentication

Selecting the `OAUTHBEARER` SASL mechanism provides the options listed in this section.

Microsoft Entra ID authentication endpoint: Specifies the Microsoft Entra ID endpoint from which to acquire authentication tokens. Defaults to `https://login.microsoftonline.com`. You can instead select `https://login.microsoftonline.us` or `https://login.partner.microsoftonline.cn`.

Client ID: Enter the `client_id` to pass in the OAuth request parameter.

Tenant identifier: Enter your Microsoft Entra ID subscription's directory ID (tenant ID).

Scope: Enter the scope to pass in the OAuth request parameter. This will be of the form: `https://<Bootstrap-Server-Hostname>/.default`. For example, for a bootstrap server `myfabricserver.example.com:9093`, **Scope** would be `https://myfabricserver.example.com/.default`.

Authentication method: Use the buttons to select one of these options:

- **Secret**: Exposes a **Client secret (text secret)** drop-down, in which you can select a stored secret that references the `client_secret`. A **Create** link is available to define a new secret.
- **Certificate**: Exposes a **Certificate** drop-down, in which you can select a stored certificate. A **Create** link is available to define a new cert.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_group` - Cribl Edge Fleet that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_mode` - Distributed mode of the instance that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Retries

These settings provide flexibility in handling retries for failed messages, allowing you to balance between quick retries and avoiding excessive load on the system.

The default configuration starts with a 300 milliseconds retry interval, doubles the interval after each retry, and caps the maximum retry interval at 30 seconds. The system will attempt to retry the request up to five times before considering it a failure.

Retry limit: Maximum number of times to retry a failed request before the message fails. Defaults to `5`. Enter `0` to not retry at all. For example, if set to `5`, the system will attempt to retry the request up to five times before considering it a failure.

Initial retry interval (ms): Initial value used to calculate the retry interval, in milliseconds. This value determines the starting point for the retry delay. The default (and minimum) is `300` ms (three seconds). The maximum allowed value is `600000` ms (10 minutes). For example, if set to `1000` ms, the first retry will occur after one second.

Backoff multiplier: Set the backoff multiplier (`2`–`20`) to control the retry frequency for failed messages. The multiplier is used in an exponential backoff formula to increase the delay between retries. For faster retries, use a lower multiplier. For slower retries with more delay between attempts, use a higher multiplier. For example, with an initial retry interval of `1000` ms and a multiplier of `2`, the retry intervals will be 1,000 ms, 2,000 ms, 4,000 ms, and so on. See the Kafka documentation for details.

Backoff limit (ms): The maximum wait time for a retry, in milliseconds. This setting caps the exponential backoff delay to prevent excessively long wait times. The default (and minimum) value is `30000` ms (30 seconds), and the maximum is `180000` ms (180 seconds). For example, if the calculated retry interval exceeds 180,000 ms, the retry will occur after 180,000 ms instead.

Advanced Settings

Record size limit (KB, uncompressed): Maximum size (KB) of each record batch before compression. Setting should be < `message.max.bytes` settings in bootstrap server. Defaults to `768`.

Events-per-batch limit: Maximum number of events in a batch before forcing a flush. Defaults to `1000`.

Flush period (sec): Maximum time between requests. Low settings could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Connection timeout (ms): Maximum time to wait for a successful connection. Defaults to `10000` ms (10 seconds). Valid range is `1000` to `3600000` ms (1 second to 1 hour).

Request timeout (ms): Maximum time to wait for Kafka to respond to a successful request. Defaults to `60000` ms (1 minute).

Authentication timeout (ms): Maximum time to wait for Kafka to respond to an authentication request. Defaults to `10000` (10 seconds).

Reauthentication threshold (ms): Time window during which Cribl Edge can reautheticate if needed, measured backward from the moment when credentials are set to expire. Defaults to `10000` (10 seconds).

A small value for this setting, combined with high network latency, might prevent the Destination from reauthenticating before the bootstrap server closes the connection.

A large value might cause the Destination to send reauthentication messages too soon, wasting bandwidth.

The Kafka setting `connections.max.reauth.ms` controls the reuthentication threshold on the Kafka side.

Working with Event Timestamps

By default, when an incoming event reaches this Destination, the underlying Kafka JS library adds a `timestamp` field set to the current time at that moment. The library gets the current time from the `Date.now()` JavaScript method. Events that this Destination sends to downstream Kafka receivers include this `timestamp` field.

Some production situations require the `timestamp` value to be the time an incoming event was originally created by an upstream sender, not the current time of its arrival at this Destination. For example, suppose the events were originally created over a wide time range, and arrive at the Destination hours or days later. In this case, the original creation time might be important to retain in events Cribl Edge sends to downstream Kafka receivers.

Here is how to satisfy such a requirement:

1. In a Pipeline that routes events to this Destination, use an Eval Function to add a field named `__kafkaTime` and write the incoming event's original creation time into that field. The value of `__kafkaTime` **must** be in Unix time (either seconds or milliseconds - both will work). For context, see this explanation of the related fields `_time` and `__origTime`.
2. This Destination will recognize the `__kafkaTime` field and write its value into the `timestamp` field. This is similar to the way you can use the `__topicOut` field to overwrite a topic setting, as described above.

If the `__kafkaTime` field is **not** present, the Destination will apply the default behavior (using `Date.now()`) described previously.

Fabric Workspace Permissions

To add a **Cribl** data source in Microsoft Fabric and retrieve the values necessary to configure the Microsoft Fabric Real-Time Intelligence Destination in Cribl Edge, you must have **Contributor** (or higher) Fabric workspace permissions.

To use `OAUTHBEARER` authentication, you need **Member** (or higher) Microsoft Fabric workspace permissions so that you can add a service principal app to the workspace with **Contributor** (or higher) permissions.

See Permission model in the official Microsoft documentation for more information.

Internal Fields

Cribl Edge uses a set of internal fields to assist in forwarding data to a Destination.

Fields for this Destination:

- `__topicOut`
- `__key`
- `__headers`
- `__keySchemaIdOut`
- `__valueSchemaIdOut`

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

12.3.23. Filesystem/NFS Destination

Filesystem is a non-streaming Destination type that Cribl Edge can use to output files to a local file system or a network-attached file system (NFS).

Type: Non-Streaming | TLS Support: N/A | PQ Support: N/A

In Cribl.Cloud, the Filesystem Destination is only available on customer-managed hybrid Edge Nodes.

Configure Cribl Edge to Output to Filesystem Destinations

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this Filesystem definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
    - **Description**: Optionally, enter a description.
    - **Output location**: Final destination for the output files.
    - **Data format**: The output data format defaults to `JSON`. `Raw` and `Parquet` are also available. Selecting `Parquet` (supported only on Linux, not Windows) exposes a **Parquet Settings** left tab, where you **must** configure certain options in order to export data in Parquet format.
3. Next, you can configure the following **Optional Settings**:
    - **Staging location**: Local filesystem location in which to buffer files before compressing and moving them to the final destination. Cribl recommends that this location be stable and high-performance.

        The **Staging location** field is not displayed or available on Cribl.Cloud-managed Edge Nodes.

- **Partitioning expression**: JavaScript expression that defines how files are partitioned and organized. Default is date-based. If blank, Cribl Edge will fall back to the event's `__partition` field value (if present); or otherwise to the root directory of the **Output Location** and **Staging Location**.
- **Compression**: Data compression format used before moving to final destination. Defaults to `gzip` (recommended). This setting is not available when **Data format** is set to `Parquet`.
- **File name prefix expression**: The output filename prefix. Must be a JavaScript expression (which can evaluate to a constant), enclosed in quotes or backticks. Defaults to `CriblOut`.
- **File name suffix expression**: The output filename suffix. Must be a JavaScript expression (which can evaluate to a constant), enclosed in quotes or backticks. Defaults to `` `.${C.env["CRIBL_WORKER_ID"]}.${__format}${__compression === "gzip" ? ".gz" : ""}` ``, where `__format` can be `json` or `raw`, and `__compression` can be `none` or `gzip`.

  To prevent files from being overwritten, Cribl appends a random sequence of 6 characters to the end of their names. This also applies to prefix and suffix expressions in file names.

  For example, if you set the **File name prefix expression** to `CriblExec` and set the **File name suffix expression** to `.csv`, the file name might display as `CriblExec-adPRWM.csv` with `adPRWM` appended.

- **Backpressure Behavior**: Select whether to block or drop events when all receivers are exerting backpressure. (Causes might include an accumulation of too many files needing to be closed.) Defaults to `Block`.
- **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

4. Optionally, you can adjust the Processing, Parquet, and Advanced settings outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.

- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Parquet Settings

To write out Parquet files, note that:

- On Linux, you can use the Cribl Edge CLI's `parquet` command to view a Parquet file, its metadata, or its schema.
- Cribl Edge Workers support Parquet only when running on Linux, not on Windows.
- See Parquet Schemas for pointers on how to avoid problems such as data mismatches.

Automatic schema: Toggle on to automatically generate a Parquet schema based on the events of each Parquet file that Cribl Edge writes. When toggled off (the default), exposes the following additional field:

- **Parquet schema**: Select a schema from the drop-down.



If you need to modify a schema or add a new one, follow the instructions in our Parquet Schemas topic. These steps will propagate the freshest schema back to this drop-down.

Parquet version: Determines which data types are supported, and how they are represented. Defaults to `2.6`; `2.4` and `1.0` are also available.

Data page version: Serialization format for data pages. Defaults to `V2`. If your toolchain includes a Parquet reader that does not support `V2`, use `V1`.

Group row limit: The number of rows that every group will contain. The final group can contain a smaller number of rows. Defaults to `10000`.

Page size: Set the target memory size for page segments. Generally, set lower values to improve reading speed, or set higher values to improve compression. Value must be a positive integer smaller than the **Row group size** value, with appropriate units. Defaults to `1 MB`.

Log invalid rows: Toggle on to output up to 20 unique rows that were skipped due to data format mismatch. Log level must be set to `debug` for output to be visible.

Write statistics: Toggle on (default) if you have Parquet tools configured to view statistics - these profile an entire file in terms of minimum/maximum values within data, numbers of nulls, etc.

Write page indexes: Toggle on (default) if your Parquet reader uses statistics from Page Indexes to enable page skipping. One Page Index contains statistics for one data page.

Write page checksum: Toggle on if you have configured Parquet tools to verify data integrity using the checksums of Parquet pages.

Metadata (optional): The metadata of files the Destination writes will include the properties you add here as key-value pairs. For example, one way to tag events as belonging to the OCSF category for security findings would be to set **Key** to `OCSF Event Class` and **Value** to `2001`.

Advanced Settings

File size limit (MB): Maximum uncompressed output file size. Files of this size will be closed and moved to final output location. Defaults to `32`.

File open time limit (sec): Maximum amount of time to write to a file. Files open for longer than this will be closed and moved to final output location. Defaults to `300`.

Idle time limit (sec): Maximum amount of time to keep inactive files open. Files open for longer than this will be closed and moved to final output location. Defaults to `30`.

Open file limit: Maximum number of files to keep open concurrently. When exceeded, the oldest open files will be closed and moved to final output location. Defaults to `100`.



Cribl Edge will close files when **either** of the `File size limit (MB)` or the `File open time limit (sec)` conditions are met.

Compression level: Specifies the level of compression to apply before moving files to the final destination. Higher compression levels reduce file size but increase CPU usage. Choose from `Best Speed`, `Normal`, or `Best Compression`. The default level, `Best Speed`, will prioritize faster compression over smaller file sizes.

Writing high watermark (KB): Sets the high watermark for writing files, in kilobytes. This determines the buffer size for file writes, impacting performance and memory usage. Enter a numeric value representing the buffer size in kilobytes. The default size, `64` will use a 64 KB buffer for file writes.

Header line: If set, this line will be written to the beginning of each output file, followed by a newline character. This can be useful for adding a header row to CSV files.

Disk space protection: Specifies whether to `Block` (default) or `Drop` incoming events when the disk space falls below the globally defined Min free disk space amount.

Add Output ID: When toggled on (default), adds the **Output ID** field's value to the staging location's file path. This ensures that each Destination's logs will write to its own bucket.



For a Destination originally configured in a Cribl Edge version below 2.4.0, the **Add Output ID** behavior will be switched **off** on the backend, regardless of this toggle's state. This is so that upon Cribl Edge upgrade and restart, any files pending in the original staging directory will not be lost. To enable this option for such Destinations, Cribl's recommended migration path is:

- Clone the Destination.

- Where Routes reference the original Destination, redirect them to instead reference the new, cloned Destination.

This way, the original Destination will process pending files (after an idle timeout), and the new, cloned Destination will process newly arriving events with **Add output ID** enabled.

Remove empty staging directories: When toggled on (the default), Cribl Edge deletes empty staging directories after moving files. This prevents the proliferation of orphaned empty directories. When enabled, exposes this additional option:

- **Staging cleanup period**: How often (in seconds) to delete empty directories when **Remove staging dirs** is enabled. Defaults to `300` seconds (every 5 minutes). Minimum configurable interval is `10` seconds; maximum is `86400` seconds (every 24 hours).
- **Directory batch size**: Specifies how many directories are scanned per batch during automatic cleanup of empty staging directories. Set between `10` and `10000`, defaults to `1000`. Higher values speed up cleanup but increase memory usage.

Force close on shutdown: Toggle on to force all staged files to close during an orderly shutdown. This triggers immediate upload or finalization of any in-progress files, regardless of **Idle time**, **File open time**, or **File size** limits. Use this option to minimize data loss risk in ephemeral or auto-scaling environments. This could result in a higher number of small files, as files are closed before reaching their configured thresholds. Ensure your Worker Group or Fleet **Drain timeout** is set high enough to allow all staged data to be finalized before shutdown completes.

Enable dead-lettering: Toggle on to set a maximum number of retries, and to move files to a designated directory when write failures exceed that limit. This prevents data flow blockage and excessive error logging due to undeliverable files. When enabled, exposes two additional fields:

- **Dead-letter location**: Specify the storage location for undeliverable files. Defaults to `$CRIBL_HOME/state/outputs/dead-letter`.
- **Maximum retry limit**: Configure the retry limit for failed file deliveries. This setting defines how many times the system will attempt to move a file to its intended location before it is deemed undeliverable and placed in the dead-letter directory. Defaults to `20`.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Internal Fields

Cribl Edge uses a set of internal fields to assist in forwarding data to a Destination.

Field for this Destination:

- `__partition`



To export events from an intermediate stage **within a Pipeline** to a file, see the Tee Function.

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

12.3.24. Honeycomb Destination

Cribl Edge supports sending events to a Honeycomb dataset.



Type: Streaming | TLS Support: Yes | PQ Support: Yes

Configure Cribl Edge to Output to Honeycomb

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this Honeycomb definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
    - **Description**: Optionally, enter a description.
    - **Dataset name**: Name of the dataset to send events to. (For example, `iLoveObservabilityDataset`.)
3. Under **Authentication**, select an **Authentication method** from the dropdown:
    - **Manual**: Displays a field for you to enter the **API key** for the team to which the dataset belongs.
    - **Secret**: This option exposes an **API key (text secret)** drop-down, in which you can select a stored secret that references the API key described above. A **Create** link is available to store a new, reusable secret.
    - **API key**: Enter your Honeycomb organization's API key.
4. Next, you can configure the following **Optional Settings**:
    - **Backpressure behavior**: Select whether to block, drop, or queue events when all receivers are exerting backpressure. (Causes might include a broken or denied connection, or a rate limiter.) Defaults to `Block`.
    - **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary)

tag names.

5. Optionally, you can adjust the Persistent Queue, Processing, Retries, and Advanced settings outlined in the sections below.
6. Select **Save**, then **Commit & Deploy**.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.



If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older

than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5` GB. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.

Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Retries

Honor Retry-After header: Toggle on to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream/Edge limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. Any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. Toggle off to ignore all `Retry-After` headers.

Default Retry Behavior: By default, Cribl Stream/Edge applies predefined retry strategies for certain failed requests to the Destination using an exponential backoff algorithm. You can override these settings by manually configuring the retry behavior for a particular error code.

Destinations created before the 4.16.0 release do not support the default `4xx` series retry behavior. If your Destination was created prior to this release and you'd like it to support the behavior, reconfigure the Destination.

To retry requests, Cribl Stream/Edge applies the following rules by default:

Status Code	Action
Any in the `1xx`, `3xx` series	Drop the request

Status Code	Action
`401`, `403`, `408`, `429` errors	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached
Any other `4xx` series	Drop the request
Any in the `5xx` series	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached

By default, any `429 (Too Many Requests)` response that is not listed in the retry settings is treated as a non-retryable `4xx` error and dropped. If you want `429` responses retried, ensure `429` is included in the **Settings for failed HTTP requests**.

Settings for failed HTTP requests: You can also manually configure Cribl Stream/Edge to automatically retry requests that receive particular HTTP response status codes.

Upon receiving a response code that's on the list, Cribl Stream/Edge first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream/Edge continues retrying the request without increasing the time between retries any further.

If the sender (which manages the connection to the Destination) is at capacity, it will not accept any incoming events. These incoming events originate internally from a previous stage of the data flow when Destinations send outbound requests to their respective external services, and they include retry requests and new requests. Any events that were already in transit when the sender reached capacity will continue to be processed downstream.

Sender capacity is freed up when an outgoing request succeeds or encounters a non-retryable error. When the sender has available capacity again, it will resume accepting incoming events. This capacity management is influenced by the number of active connections and configured limits, such as concurrency and buffer sizes. If a Pipeline sends events faster than the Destination can process, the buffers may fill up, leading to backpressure and `Sender at capacity` warnings. This backpressure prevents the sender from accepting additional requests until capacity is restored.

For each response code you want to add to the list, select **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.

- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: Toggle on to automatically retry requests that have timed out and display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Toggle on to reject certificates that are **not** authorized by a CA in the **CA certificate path**, nor by another trusted CA (for example, the system's CA).

Round-robin DNS: Toggle on to enable round-robin DNS lookup across multiple IP addresses, IPv4 and IPv6. When a DNS server resolves a Fully Qualified Domain Name (FQDN) to multiple IP addresses, Cribl Edge will sequentially use each address in the order they are returned by the DNS server for subsequent connection attempts.

Compress: Toggle on (default) to compress the payload body before sending.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests per Worker Process. When Cribl Edge hits this limit, it begins throttling traffic to the downstream service. Defaults to `5`. Minimum: `1`. Maximum: `32`.

Body size limit (KB): Maximum size of the request body before compression. Defaults to `4096` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Body size limit** value until downstream receivers reliably accept all events.

Events-per-request limit: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Extra HTTP headers: Name-value pairs to pass as additional HTTP headers. Values will be sent encrypted.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Cribl Edge will redact all headers, except non-sensitive headers that you declare below in **Safe headers**.

Safe headers: Add headers to declare them as safe to log in plaintext. (Sensitive headers such as `authorization` will always be redacted, even if listed here.) Use a tab or hard return to separate header names.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Notes on HTTP-Based Outputs

- To proxy outbound HTTP/S requests, see System Proxy Configuration.
- Cribl Edge will attempt to use keepalives to reuse a connection for multiple requests. After two minutes of the first use, the connection will be thrown away, and a new one will be reattempted. This is to prevent sticking to a particular Destination when there is a constant flow of events.
- If the server does not support keepalives (or if the server closes a pooled connection while idle), a new connection will be established for the next request.
- When resolving the Destination's hostname, Cribl Edge will pick the first IP in the list for use in the next connection. Enable Round-robin DNS to better balance distribution of events between destination cluster nodes.

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

12.3.25. InfluxDB Destination

Cribl Edge supports sending data to InfluxDB (versions 1.x and 2.0.x) and InfluxDB Cloud.



Type: Streaming | TLS Support: Configurable | PQ Support: Yes

Configure Cribl Edge to Output to InfluxDB

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this InfluxDB definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
    - **Description**: Optionally, enter a description.
    - **Write API URL**: The URL of an InfluxDB cluster to send events to:
        - v1 API: `http://localhost:8086/write`
        - v2 API: `http://localhost:8086/api/v2/write`
    - **Use v2 API**: You can enable the InfluxDB v2 API with InfluxDB version 1.8 or later. Toggle off (default) to fall back to the v1 API and display a **Database** field. If you toggle **Use v2 API** on, Cribl Edge communicates using InfluxDB's v2 API, and instead displays these two fields:
        - **Bucket**: Enter the bucket to write to (required).
        - **Organization**: The Organization ID corresponding to the specified **Bucket** (required in this configuration, although InfluxDB v.1.8 will ignore it).
    - **Database**: Name of the database on which to write data points. Displayed only for configurations that use the v1 API (required).
3. Next, you can configure the following **Optional Settings**:

- **Timestamp precision**: Sets the precision for the supplied Unix time values. Defaults to `Milliseconds`.
- **Dynamic value fields**: When toggled on (default), Cribl Edge will pull the value field from the metric name. For example, `db.query.user` will use `db.query` as the measurement and `user` as the value field.
- **Value field name**: Name of the field in which to store the metric when sending to InfluxDB. This will be used as a fallback if dynamic name generation is enabled but fails. Defaults to `value`.
- **Backpressure behavior**: Select whether to block, drop, or queue events when all receivers are exerting backpressure. (Causes might include a broken or denied connection, or a rate limiter.) Defaults to `Block`.
- **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

4. Optionally, you can adjust the Authentication, Persistent Queue, Processing, Retries, and Advanced settings outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

Authentication

Use the **Authentication type** drop-down to select one of these options:

None: This default setting does not use authentication.

Auth token: Use HTTP token authentication. In the resulting **Token** field, enter the bearer token that must be included in the HTTP authorization header.

Auth token (text secret): This option exposes a **Token (text secret)** drop-down, in which you can select a stored text secret that references the bearer token described above. A **Create** link is available to store a new, reusable secret.

Basic: Displays **Username** and **Password** fields for you to enter HTTP Basic authentication credentials.

Basic (credentials secret): This option exposes a **Credentials secret** drop-down, in which you can select a stored text secret that references the Basic authentication credentials described above. A **Create** link is available to store a new, reusable secret.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.



If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5 GB`. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.



Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options

include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Retries

Honor Retry-After header: Toggle on to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream/Edge limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. Any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. Toggle off to ignore all `Retry-After` headers.

Default Retry Behavior: By default, Cribl Stream/Edge applies predefined retry strategies for certain failed requests to the Destination using an exponential backoff algorithm. You can override these settings by manually configuring the retry behavior for a particular error code.



Destinations created before the 4.16.0 release do not support the default `4xx` series retry behavior. If your Destination was created prior to this release and you'd like it to support the behavior, reconfigure the Destination.

To retry requests, Cribl Stream/Edge applies the following rules by default:

Status Code	Action
Any in the `1xx`, `3xx` series	Drop the request
`401`, `403`, `408`, `429` errors	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached
Any other `4xx` series	Drop the request
Any in the `5xx` series	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached

By default, any `429 (Too Many Requests)` response that is not listed in the retry settings is treated as a non-retryable `4xx` error and dropped. If you want `429` responses retried, ensure `429` is included in the **Settings for failed HTTP requests**.

Settings for failed HTTP requests: You can also manually configure Cribl Stream/Edge to automatically retry requests that receive particular HTTP response status codes.

Upon receiving a response code that's on the list, Cribl Stream/Edge first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream/Edge continues retrying the request without increasing the time between retries any further.

If the sender (which manages the connection to the Destination) is at capacity, it will not accept any incoming events. These incoming events originate internally from a previous stage of the data flow when Destinations send outbound requests to their respective external services, and they include retry requests and new requests. Any events that were already in transit when the sender reached capacity will continue to be processed downstream.

Sender capacity is freed up when an outgoing request succeeds or encounters a non-retryable error. When the sender has available capacity again, it will resume accepting incoming events. This capacity management is influenced by the number of active connections and configured limits, such as concurrency and buffer sizes. If a Pipeline sends events faster than the Destination can process, the buffers may fill up, leading to backpressure and `Sender at capacity` warnings. This backpressure prevents the sender from accepting additional requests until capacity is restored.

For each response code you want to add to the list, select **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: Toggle on to automatically retry requests that have timed out and display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3

minutes).

Advanced Settings

Validate server certs: Reject certificates that are **not** authorized by a trusted CA (for example, the system's CA). Defaults to toggled on.

Round-robin DNS: Toggle on to enable round-robin DNS lookup across multiple IP addresses, IPv4 and IPv6. When a DNS server resolves a Fully Qualified Domain Name (FQDN) to multiple IP addresses, Cribl Edge will sequentially use each address in the order they are returned by the DNS server for subsequent connection attempts.

Compress: Toggle on (default) to compress the payload body before sending.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests before blocking. This is set per Worker Process. Defaults to `5`.

Body size limit (KB): Maximum size of the request body before compression. Defaults to `4096` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Body size limit** value until downstream receivers reliably accept all events.

Events-per-request limit: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Extra HTTP headers: Name-value pairs to pass as additional HTTP headers. Values will be sent encrypted.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Cribl Edge will redact all headers, except non-sensitive headers that you declare below in **Safe headers**.

Safe headers: Add headers to declare them as safe to log in plaintext. (Sensitive headers such as `authorization` will always be redacted, even if listed here.) Use a tab or hard return to separate header names.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate,

12.3.26. MinIO Destination

MinIO is a non-streaming Destination type, to which Cribl Edge can output objects.



Type: Non-Streaming | TLS Support: Configurable | PQ Support: No

Configure Cribl Edge to Output to MinIO Destinations

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this MinIO definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
    - **Description**: Optionally, enter a description.
    - **MinIO endpoint**: MinIO service URL (for example, http://minioHost:9000).
    - **MinIO bucket name**:Name of the destination MinIO bucket. This value can be a constant, or a JavaScript expression that will be evaluated only at init time. For example, referencing a Global Variable: `myBucket-${C.vars.myVar}`. Ensure that the bucket already exists, otherwise MinIO will generate "bucket does not exist" errors.

      Event-level variables are not available for JavaScript expressions. This is because the bucket name is evaluated only at Destination initialization. If you want to use event-level variables in file paths, Cribl recommends specifying them in the **Partitioning Expression** field (described below), because this is evaluated for each file.
    - **Staging location**: Filesystem location in which to locally buffer files before compressing and moving to final destination. Cribl recommends that this location be stable and high-performance.

      The **Staging location** field is not displayed or available on Cribl.Cloud-managed Edge Nodes.

- **Data format**: The output data format defaults to `JSON`. `Raw` and `Parquet` are also available. Selecting `Parquet` (supported only on Linux, not Windows) exposes a **Parquet Settings** left tab, where you **must** configure certain options in order to export data in Parquet format.

3. Next, you can configure the following **Optional Settings**:
   - **Key prefix**: Root directory to prepend to path before uploading. Enter a constant, or a JS expression enclosed in single quotes, double quotes, or backticks. Prefix to apply to files/objects before uploading to the specified bucket. MinIO will display key prefixes as folders.
   - **Partitioning expression**: JavaScript expression that defines how files are partitioned and organized. Default is date-based. If blank, Cribl Edge will fall back to the event's `__partition` field value (if present); or otherwise to the root directory of the **Output Location** and **Staging Location**.

     Cribl Edge's internal `__partition` field can be populated in multiple ways. The precedence order is: explicit **Partitioning expression** value -> `${host}/${sourcetype}` (default) **Partitioning expression** value -> user-defined `event.__partition`, set with an Eval Function (takes effect only where this **Partitioning expression** field is blank).

   - **Compression**: Data compression format used before moving to final destination. Defaults to `gzip` (recommended). This setting is not available when **Data format** is set to `Parquet`.
   - **File name prefix expression**: The output filename prefix. Must be a JavaScript expression (which can evaluate to a constant), enclosed in quotes or backticks. Defaults to `CriblOut`.
   - **File name suffix expression**: The output filename suffix. Must be a JavaScript expression (which can evaluate to a constant), enclosed in quotes or backticks. Defaults to `` `.${C.env["CRIBL_WORKER_ID"]}.${__format}${__compression === "gzip" ? ".gz" : ""}` ``, where `__format` can be `json` or `raw`, and `__compression` can be `none` or `gzip`.
   - **Backpressure behavior**: Select whether to block or drop events when all receivers are exerting backpressure. (Causes might include an accumulation of too many files needing to be closed.) Defaults to `Block`.
   - **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

4. Optionally, you can adjust the Authentication, Processing, Parquet, and Advanced settings outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

How MinIO Composes File Names

The full path to a file consists of:

`<bucket_name>/<keyprefix><partition_expression | __partition><file_name_prefix><filename>.<extension>`

As an example, assume that the **MinIO bucket name** is `bucket1`, the **Key prefix** is `aws`, the **Partitioning expression** is `` `${host}/${sourcetype}` ``, the source is undefined, the **File name prefix** is the default `CriblOut`, and the **Data format** is `json`. Here, the full path as displayed in MinIO would have this form: `/bucket1/aws/192.168.1.241/undefined/CriblOut-<randomstring>0.json`



Although MinIO will display the **Key prefix** and **Partitioning expression** values as folders, both are actually just part of the overall key name, along with the file name.

Authentication

Use the **Authentication Method** drop-down to select one of these options:

Auto: This default option uses the AWS SDK for JavaScript to automatically obtain credentials in the following order of attempts:

1. **Environment Variables**: Loaded from environment variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`.
2. **IAM Identity Center (SSO)**: Loaded from AWS IAM Identity Center configuration.
3. **Shared Credentials File**: Loaded from the shared credentials file (`~/.aws/credentials`).
4. **IAM Roles for Amazon EC2/ECS**: Loaded from AWS Identity and Access Management (IAM) roles attached to an EC2 instance or ECS container.
5. **JSON File on Disk**: Loaded from a JSON file on disk.
6. **Other Credential-Provider Classes**: Other credential-provider classes provided by the AWS SDK for JavaScript.

The `Auto` method works both when running on AWS and in other environments where the necessary credentials are available through one of the above methods.



SSO Providers

When using the auto authentication method, you can leverage SSO providers like SAML and Okta to issue temporary credentials. These credentials should be set in the environment variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`. The AWS SDK will then use these environment variables to authenticate.

Manual: If not running on AWS, you can select this option to enter a static set of user-associated IAM credentials (your access key and secret key) directly or by reference. This is useful for Workers not in an AWS

VPC, for example, those running a private cloud. The **Manual** option exposes these corresponding additional fields:

- **Access key**: Enter your AWS access key. If not present, will fall back to the `env.AWS_ACCESS_KEY_ID` environment variable, or to the metadata endpoint for IAM role credentials.
- **Secret key**: Enter your AWS secret key. If not present, will fall back to the `env.AWS_SECRET_ACCESS_KEY` environment variable, or to the metadata endpoint for IAM credentials.

The values for **Access key** and **Secret key** can be a constant, or a JavaScript expression (such as `${C.env.MY_VAR}`) enclosed in quotes or backticks, which allows configuration with environment variables.

Secret: If not running on AWS, you can select this option to supply a stored secret that references an AWS access key and secret key. This option exposes a **Secret key pair** drop-down, in which you can select a stored secret that references the set of user-associated IAM credentials described above. A **Create** link is available to store a new, reusable secret.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Parquet Settings

To write out Parquet files, note that:

- On Linux, you can use the Cribl Edge CLI's `parquet` command to view a Parquet file, its metadata, or its schema.
- Cribl Edge Workers support Parquet only when running on Linux, not on Windows.
- See Parquet Schemas for pointers on how to avoid problems such as data mismatches.

Automatic schema: Toggle on to automatically generate a Parquet schema based on the events of each Parquet file that Cribl Edge writes. When toggled off (the default), exposes the following additional field:

- **Parquet schema**: Select a schema from the drop-down.



If you need to modify a schema or add a new one, follow the instructions in our Parquet Schemas topic. These steps will propagate the freshest schema back to this drop-down.

Parquet version: Determines which data types are supported, and how they are represented. Defaults to `2.6`; `2.4` and `1.0` are also available.

Data page version: Serialization format for data pages. Defaults to `V2`. If your toolchain includes a Parquet reader that does not support `V2`, use `V1`.

Group row limit: The number of rows that every group will contain. The final group can contain a smaller number of rows. Defaults to `10000`.

Page size: Set the target memory size for page segments. Generally, set lower values to improve reading speed, or set higher values to improve compression. Value must be a positive integer smaller than the **Row group size** value, with appropriate units. Defaults to `1 MB`.

Log invalid rows: Toggle on to output up to 20 unique rows that were skipped due to data format mismatch. Log level must be set to `debug` for output to be visible.

Write statistics: Leave toggled on (the default) if you have Parquet tools configured to view statistics - these profile an entire file in terms of minimum/maximum values within data, numbers of nulls, etc.

Write page indexes: Leave toggled on (the default) if your Parquet reader uses statistics from Page Indexes to enable page skipping. One Page Index contains statistics for one data page.

Write page checksum: Toggle on if you have configured Parquet tools to verify data integrity using the checksums of Parquet pages.

Metadata (optional): The metadata of files the Destination writes will include the properties you add here as key-value pairs. For example, one way to tag events as belonging to the OCSF category for security findings would be to set **Key** to `OCSF Event Class` and **Value** to `2001`.

Advanced Settings

File size limit (MB): Maximum uncompressed output file size. Files of this size will be closed and moved to final output location. Defaults to `32`.

File open time limit (sec): Maximum amount of time to write to a file. Files open for longer than this limit will be closed and moved to final output location. Defaults to `300`.

Idle time limit (sec): Maximum amount of time to keep inactive files open. Files open for longer than this limit will be closed and moved to final output location. Defaults to `30`.

Open file limit: Maximum number of files to keep open concurrently. When exceeded, the oldest open files will be closed and moved to final output location. Defaults to `100`.



Cribl Edge will close files when **either** of the `File size limit (MB)` or the `File open time limit (sec)` conditions is met.

Concurrent file parts limit: Maximum number of parts to upload in parallel per file. A value of `1` tells the Destination to send the whole file at once. When set to `2` or above, IAM permissions must include those required for multipart uploads. Defaults to `4`; highest allowed value is `10`.

Disk space protection: Specifies whether to `Block` (default) or `Drop` incoming events when the disk space falls below the globally defined Min free disk space amount.

Add Output ID: When toggled on (default), adds the **Output ID** field's value to the staging location's file path. This ensures that each Destination's logs will write to its own bucket.



For a Destination originally configured in a Cribl Edge version below 2.4.0, the **Add Output ID** behavior will be switched **off** on the backend, regardless of this toggle's state. This is to avoid losing any files pending in the original staging directory, upon Cribl Edge upgrade and restart. To enable this option for such Destinations, Cribl's recommended migration path is:

- Clone the Destination.
- Redirect the Routes referencing the original Destination to instead reference the new, cloned Destination.

This way, the original Destination will process pending files (after an idle timeout), and the new, cloned Destination will process newly arriving events with **Add output ID** enabled.

Remove empty staging directories: When toggled on (the default), Cribl Edge deletes empty staging directories after moving files. This prevents the proliferation of orphaned empty directories. When enabled, exposes this additional option:

- **Staging cleanup period**: How often (in seconds) to delete empty directories when **Remove staging dirs** is enabled. Defaults to `300` seconds (every 5 minutes). Minimum configurable interval is `10` seconds; maximum is `86400` seconds (every 24 hours).
- **Directory batch size**: Specifies how many directories are scanned per batch during automatic cleanup of empty staging directories. Set between `10` and `10000`, defaults to `1000`. Higher values speed up cleanup but increase memory usage.

Enable dead-lettering: Toggle this on to set a maximum number of retries, and to move files to a designated directory when write failures exceed that limit. This prevents data flow blockage and excessive error logging due to undeliverable files. When enabled, exposes two additional fields:

- **Dead-letter location**: Specify the storage location for undeliverable files. Defaults to `$CRIBL_HOME/state/outputs/dead-letter`.
- **Maximum retry limit**: Configure the retry limit for failed file deliveries. This setting defines how many times the system will attempt to move a file to its intended location before it is deemed undeliverable and placed in the dead-letter directory. Defaults to `20`.

Force close on shutdown: Toggle on to force all staged files to close during an orderly shutdown. This triggers immediate upload or finalization of any in-progress files, regardless of **Idle time**, **File open time**, or **File size** limits. Use this option to minimize data loss risk in ephemeral or auto-scaling environments. This could result in a higher number of small files, as files are closed before reaching their configured thresholds. Ensure your Worker Group or Fleet **Drain timeout** is set high enough to allow all staged data to be finalized before shutdown completes.

Region: Region where the MinIO service/cluster is located. Leave blank when using a containerized MinIO.

Object ACL: ACL (Access Control List) to assign to uploaded objects. Defaults to `Private`.

Storage class: Select a storage class for uploaded objects. Defaults to `Standard`.

Server-side encryption: Server side encryption type for uploaded objects. Defaults to `none`.

Signature version: Signature version to use for signing MinIO requests. Defaults to `v4`.

Reuse connections: Whether to reuse connections between requests. Toggling on (default) can improve performance.

Reject unauthorized certificates: Whether to accept certificates that cannot be verified against a valid Certificate Authority (for example, self-signed certificates). Defaults to toggled on.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

IAM Permissions

The following permissions are always needed to write to an Amazon S3-compatible object store:

- `s3:ListBucket`
- `s3:GetBucketLocation`
- `s3:PutObject`

If your Destination needs to do multipart uploads to S3, two more permissions are needed:

- `kms:GenerateDataKey`
- `kms:Decrypt`

See the AWS documentation.

Internal Fields

Cribl Edge uses a set of internal fields to assist in forwarding data to a Destination.

Field for this Destination:

- `__partition`

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

Common Issues

"Parse Error: Expected HTTP/"

The Worker is trying to use HTTP, but the server is expecting HTTPS.



Parse Error: Expected HTTP

“Bucket does not exist - self signed certificate”

Example Error Text:

```
message: Bucket does not exist - self signed certificate
stack: Error: self signed certificate
    at TLSSocket.onConnectSecure (node:_tls_wrap_1535:34)
    at TLSSocket.emit (node:events:513:28)
    at TLSSocket.emit (node:events:489:12)
    at TLSSocket._fini... Show more
tmpPath: /opt/cribl/s3_bucket/cribl/2023/03/28/CriblOut-XTNGh5.0.json.tmp
```

Copy

Can occur when:

- the Destination uses a self-signed cert that has not yet been trusted by Cribl Stream.
- the user does not have appropriate permissions to view the bucket.
- the authentication token generated by your computer has a timestamp that is out of sync with the server’s time, resulting in an authentication failure.

Recommendation

Disable **Advanced Settings** > **Reject unauthorized certificates** or; get the sender’s cert and add it to the `NODE_EXTRA_CA_CERTS` path for validation; or verify that the user has appropriate permissions.

12.3.27. OpenTelemetry (OTel) Destination

Cribl Edge supports sending events to OpenTelemetry Protocol (OTLP)-compliant targets. (Cribl Edge can receive OTel events through the OTel Source.) Besides native OTel Trace, Metric, and Log events, you can send Cribl Edge's Gauge metric events through this OTel Destination.



Type: Streaming | TLS Support: Configurable | PQ Support: Yes

Protocol and Transport Support

This Destination supports either of the transports that OTLP specifies: gRPC or HTTP. OTLP defines Protocol buffer (Protobuf) schemas for its payloads (requests and responses). With the HTTP transport, this Destination supports Binary Protobuf payload encoding, but does not support JSON Protobuf.

When configuring Pipelines (including pre-processing and post-processing Pipelines), you need to ensure that events sent to this Destination conform to the relevant Protobuf specification. Links below are for OTLP 1.3.1:

- For traces, opentelemetry/proto/trace/v1/trace.proto
- For metrics, opentelemetry/proto/metrics/v1/metrics.proto
- For logs, opentelemetry/proto/logs/v1/logs.proto

The OTel Destination will drop non-conforming events. If the Destination encounters a parsing error when trying to convert an event to OTLP, it discards the event and Cribl Edge logs the error.



If sending data to a downstream receiver that accepts OTLP 1.3.1 log events, you must select **Advanced Settings** > **OTLP version** > `1.3.1`. Otherwise, the Destination will send log events formatted for OTLP `0.10.0`. For context, see Breaking Changes Between OTLP 0.19.0 and 1.3.1.

Configure Cribl Edge to Output to OTel

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.

- To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.

2. In the **New Destination** modal, configure the following under **General Settings**:
   - **Output ID**: Enter a unique name to identify this OTel output definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
   - **Description**: Optionally, enter a description.
   - **OTLP version**: The drop-down offers `0.10.0` (default) and `1.3.1`.
   - **Protocol**: Use the drop-down to choose the protocol to use when sending data: `gRPC` (the default), or `HTTP`. Additional settings display in the Advanced for both gRPC and HTTP.

     When set to `HTTP`, the UI adds the Retries section (left tab) and displays three additional settings below:
   - **Traces endpoint override**: By default, the Destination will send traces to `<endpoint>/v1/traces`, where `<endpoint>` is what you specified for the **Endpoint** setting above. To send traces to a different endpoint, enter that endpoint here.
   - **Metrics endpoint override**: By default, the Destination will send metrics to `<endpoint>/v1/metrics`, where `<endpoint>` is what you specified for the **Endpoint** setting above. To send metrics to a different endpoint, enter that endpoint here.
   - **Logs endpoint override**: By default, the Destination will send logs to `<endpoint>/v1/logs`, where `<endpoint>` is what you specified for the **Endpoint** setting above. To send logs to a different endpoint, enter that endpoint here.
   - **Endpoint**: Where to send events, in any of a variety of formats (FQDN, PQDN, IP address and port, etc). Supports both IPv4 and IPv6 - IPv6 addresses must be enclosed in square brackets. The same endpoint is used for both Traces and Metrics. If no port is specified, we normally default to the standard port for OTel Collectors, 4137 - however, if TLS is enabled or the endpoint is an HTTPS-based URL, we default instead to port 443.

     To proxy outbound HTTP/S requests, see System Proxy Configuration.
   - **Backpressure behavior**: Whether to block, drop, or queue events when all receivers are exerting backpressure.
   - **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
3. Optionally, you can adjust the Authentication, TLS, Persistent Queue, Processing, Retries, and Advanced settings outlined in the sections below.
4. Select **Save**, then **Commit & Deploy**.

Authentication

Select one of the following options for authentication:

- **None**: Don't use authentication.
- **Auth token**: Enter the bearer token that must be included in the authorization header. Since OpenTelemetry runs over gRPC, authorization headers are sent as Metadata entries which are essentially key-value pairs. For example: `Bearer <your-configured-token>`.
- **Auth token (text secret)**: This option exposes a drop-down in which you can select a stored text secret that references the bearer token described above. A **Create** link is available to store a new, reusable secret.
- **Basic**: This default option displays fields for you to enter HTTP Basic authentication credentials.
- **Basic (credentials secret)**: This option exposes a **Credentials secret** drop-down, in which you can select a stored text secret that references the Basic authentication credentials described above. A **Create** link is available to store a new, reusable secret.

TLS Settings (Client Side)

TLS is available only when **General Settings** > **Protocol** is set to `gRPC`.

Enabled Default is toggled off. When toggled on:

Validate server certs: Toggle on (default) to reject certificates that are **not** authorized by a CA in the **CA certificate path**, nor by another trusted CA (for example, the system's CA).

Minimum TLS version: Optionally, select the minimum TLS version to use when connecting.

Maximum TLS version: Optionally, select the maximum TLS version to use when connecting.

Certificate: Select an existing certificate name from the drop-down or add a new certificate by selecting **Create**. See Add a TLS Certificate to a Fleet for details.

CA certificate path: Path on client containing CA certificates (in PEM format) to use to verify the server's cert. Path can reference `$ENV_VARS`.

Private key path (mutual auth): Path on client containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Certificate path (mutual auth): Path on client containing certificates in (PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Passphrase: Passphrase to use to decrypt private key.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.



If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5` GB. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.



Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output - both metric events, as dimensions; and log events, as labels. Supports wildcards.

By default, includes `cribl_pipe` (Cribl Edge Pipeline that processed the event).

Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Retries



This tab is displayed when **General Settings** > **Optional Settings** > **Protocol** is set to `HTTP`.

Honor Retry-After header: Toggle on to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream/Edge limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. Any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. Toggle off to ignore all `Retry-After` headers.

Default Retry Behavior: By default, Cribl Stream/Edge applies predefined retry strategies for certain failed requests to the Destination using an exponential backoff algorithm. You can override these settings by manually configuring the retry behavior for a particular error code.



Destinations created before the 4.16.0 release do not support the default `4xx` series retry behavior. If your Destination was created prior to this release and you'd like it to support the behavior, reconfigure the Destination.

To retry requests, Cribl Stream/Edge applies the following rules by default:

Status Code	Action
Any in the `1xx`, `3xx` series	Drop the request
`401`, `403`, `408`, `429` errors	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached
Any other `4xx` series	Drop the request
Any in the `5xx` series	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached

By default, any `429 (Too Many Requests)` response that is not listed in the retry settings is treated as a non-retryable `4xx` error and dropped. If you want `429` responses retried, ensure `429` is included in the **Settings for failed HTTP requests**.

Settings for failed HTTP requests: You can also manually configure Cribl Stream/Edge to automatically retry requests that receive particular HTTP response status codes.

Upon receiving a response code that's on the list, Cribl Stream/Edge first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream/Edge continues retrying the request without increasing the time between retries any further.

If the sender (which manages the connection to the Destination) is at capacity, it will not accept any incoming events. These incoming events originate internally from a previous stage of the data flow when Destinations send outbound requests to their respective external services, and they include retry requests and new requests. Any events that were already in transit when the sender reached capacity will continue to be processed downstream.

Sender capacity is freed up when an outgoing request succeeds or encounters a non-retryable error. When the sender has available capacity again, it will resume accepting incoming events. This capacity management is influenced by the number of active connections and configured limits, such as concurrency and buffer sizes. If a Pipeline sends events faster than the Destination can process, the buffers may fill up, leading to backpressure and `Sender at capacity` warnings. This backpressure prevents the sender from accepting additional requests until capacity is restored.

For each response code you want to add to the list, select **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).

- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: Toggle on to automatically retry requests that have timed out and display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

This tab's options depend on whether **General Settings** > **Protocol** is set to the `gRPC` or `HTTP` transport. The common settings directly are displayed for both transports.

Compression: Compression type to apply to messages sent to the OpenTelemetry endpoint. `Gzip` (the default) and `None` are available for both protocols; `Deflate` is available for gRPC only.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30` sec.

Request concurrency: Maximum number of concurrent requests before blocking. This is set per Worker Process. Defaults to `5`.

Body size limit (KB): Maximum size of the request body before compression. Defaults to `4096` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Body size limit** value until downstream receivers reliably accept all events.

Flush period (sec): Maximum time between requests. Low values could cause the payload size to be smaller than the configured **Max body size**. Defaults to `1` sec.

Environment: Optionally, specify a single Git branch on which to enable this configuration. If this field is empty, the config will be enabled everywhere.

Additional Settings for gRPC

When **General Settings** > **Protocol** is set to `gRPC`, the **Advanced Settings** tab adds the following options.

Connection timeout: Amount of time (milliseconds) to wait for the connection to establish before retrying. Defaults to `10000` (10 sec.).

Keep alive time (seconds): How often the sender should ping the peer to keep the connection alive. Defaults to `30`.

Metadata: Extra information to send with each gRPC request. Click **Add Metadata** to add each item as a **Key-Value** pair. The **Key** field is arbitrary. The **Value** field is a JavaScript expression that is evaluated just once, when this Destination is initialized. If you pass credentials as metadata, Cribl recommends using C.Secret().

Additional Settings for HTTP

When **General Settings** > **Protocol** is set to `HTTP`, the **Advanced Settings** tab adds the following options.

Validate server certs: Toggle on to reject certificates that are **not** authorized by a CA in the **CA certificate path**, nor by another trusted CA (for example, the system's CA).

Round-robin DNS: Toggle on to enable round-robin DNS lookup across multiple IP addresses, IPv4 and IPv6. When a DNS server resolves a Fully Qualified Domain Name (FQDN) to multiple IP addresses, Cribl Edge will sequentially use each address in the order they are returned by the DNS server for subsequent connection attempts.

Keep alive: By default, Cribl Edge sends `Keep-Alive` headers to the remote server and preserves the connection from the client side up to a maximum of 120 seconds. Toggle this off if you want Cribl Edge to close the connection immediately after sending a request.

Extra HTTP headers: Click **Add Header** to define additional HTTP headers to pass to all events. Each row is a **Name-Value** pair. Values will be sent encrypted.

Safe headers: Add headers here to declare them as safe to log in plaintext. (Sensitive headers such as `authorization` will always be redacted, even if listed here.) Use a tab or hard return to separate header names.

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

12.3.28. SentinelOne AI SIEM Destination

The **SentinelOne AI SIEM** Destination sends data to SentinelOne's Singularity AI SIEM platform using the HTTP Event Collector (HEC) protocol.

Type: Streaming | TLS Support: Yes | PQ Support: No

Configure a SentinelOne AI SIEM Destination

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this Destination. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
    - **Description**: Optionally, enter a description.
    - **Region**: Select the geographical region for your SentinelOne AI SIEM ingestion endpoint. This must match your tenant's deployment to ensure correct data routing and compliance.
        - **US** - `United States: https://ingest.us1.sentinelone.net`
        - **CA** - `Canada: https://ingest.ca1.sentinelone.net`
        - **EMEA** - `Europe, Middle East, and Africa: https://ingest.eu1.sentinelone.net`
        - **AP** - `India: https://ingest.ap1.sentinelone.net`
        - **APS** - `India (South): https://ingest.aps1.sentinelone.net`
        - **AU** - `Australia: https://ingest.apse2.sentinelone.net`
        - **Custom Tenant**: Enter your **Base AI SIEM endpoint URL** in this format:
            - `https://<Your-S1-Tenant>.sentinelone.net`.

        - For SentinelOne DataSet (Scalyr) tenants, use the base DataSet ingest endpoint `https://ingest.scalyr.com` or `https://ingest.eu.scalyr.com` in the EU.
    - The URL must begin with `http://` or `https://`, can include a port number, must not have a trailing slash, and match this pattern: `^https?://[a-zA-Z0-9.-]+(:[0-9]+)?$`.
- **AI SIEM endpoint path**: Select the HTTP API route used to send events to SentinelOne AI SIEM. This choice determines the expected data format and how custom fields are handled.
    - `/services/collector/event` (Default): Select this path to send structured JSON payloads with standard HEC top-level fields. The **Event Fields** tab is available to configure metadata fields. Standard HEC fields are evaluated and set as top-level fields in the outgoing JSON payload. Custom metadata fields are included under the `fields` object.
    - `/services/collector/raw`: Select this path to send raw, unstructured log lines (plain text). The **Raw Fields** tab is available to configure metadata fields. The fields are sent as query parameters in the request URL.

3. Under **Authentication**, select an **Authentication method** from the dropdown:
    - **Manual**: Displays an **AI SIEM API Key** field for you to enter the API key string used to authenticate requests to the SentinelOne AI SIEM endpoint. This key is included in the `Authorization` header as a Bearer token for each HTTP request sent.
    - **Secret**: This option exposes an **AI SIEM API Key (text secret)** drop-down, in which you can select a stored secret that references the API key. A **Create** link is available to store a new, reusable secret.



This Destination does not support Mutual TLS (mTLS).

4. Next, you can configure the following **Optional Settings**:
    - **Backpressure behavior**: Select whether to block, drop, or queue events when all receivers are exerting backpressure. (Causes might include a broken or denied connection, or a rate limiter.) Defaults to `Block`.
    - **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
5. Depending on the **AI SIEM endpoint path** you selected, you can configure either **Event Fields** or **Raw Fields**. Fields have a default expression that you can adjust for dynamic data extraction or static assignment of values.



Event Fields are JavaScript expressions and **Raw Fields** are string expressions except for **serverHost** which is a JavaScript expression. In JavaScript expressions, you must enclose text

> constants in single quotes, like `'security'`. Enter string expressions directly without quotes, spaces are allowed.

Field	Event Default	Raw Default
serverHost	`__e.host || C.os.hostname()`	`C.os.hostname()`
logFile	`__e.source || (__e.__criblMetrics ? 'metrics' : 'cribl')`	`cribl`
parser	`__e.sourcetype || 'dottedJson'`	`hecRawParser`
dataSource.category	`'security'`	`security`
dataSource.name	`__e.__dataSourceName || 'cribl'`	`cribl`
dataSource.vendor	`__e.__dataSourceVendor || 'cribl'`	`cribl`
event.type		

6. Optionally, you can adjust the Processing, Retries and Advanced settings outlined in the sections below.
7. Select **Save**, then **Commit & Deploy**.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

> On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.
>
> This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.



If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5 GB`. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.



Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Retries

Honor Retry-After header: Toggle on to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream/Edge limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. Any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. Toggle off to ignore all `Retry-After` headers.

Default Retry Behavior: By default, Cribl Stream/Edge applies predefined retry strategies for certain failed requests to the Destination using an exponential backoff algorithm. You can override these settings by

manually configuring the retry behavior for a particular error code.



Destinations created before the 4.16.0 release do not support the default `4xx` series retry behavior. If your Destination was created prior to this release and you'd like it to support the behavior, reconfigure the Destination.

To retry requests, Cribl Stream/Edge applies the following rules by default:

Status Code	Action
Any in the `1xx, 3xx` series	Drop the request
`401, 403, 408, 429` errors	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached
Any other `4xx` series	Drop the request
Any in the `5xx` series	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached

By default, any `429 (Too Many Requests)` response that is not listed in the retry settings is treated as a non-retryable `4xx` error and dropped. If you want `429` responses retried, ensure `429` is included in the **Settings for failed HTTP requests**.

Settings for failed HTTP requests: You can also manually configure Cribl Stream/Edge to automatically retry requests that receive particular HTTP response status codes.

Upon receiving a response code that's on the list, Cribl Stream/Edge first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream/Edge continues retrying the request without increasing the time between retries any further.

If the sender (which manages the connection to the Destination) is at capacity, it will not accept any incoming events. These incoming events originate internally from a previous stage of the data flow when Destinations send outbound requests to their respective external services, and they include retry requests and new requests. Any events that were already in transit when the sender reached capacity will continue to be processed downstream.

Sender capacity is freed up when an outgoing request succeeds or encounters a non-retryable error. When the sender has available capacity again, it will resume accepting incoming events. This capacity management is influenced by the number of active connections and configured limits, such as concurrency and buffer sizes. If a Pipeline sends events faster than the Destination can process, the buffers may fill up, leading to backpressure

and `Sender at capacity` warnings. This backpressure prevents the sender from accepting additional requests until capacity is restored.

For each response code you want to add to the list, select **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: Toggle on to automatically retry requests that have timed out and display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Reject certificates that are not authorized by a trusted CA (for example, the system's CA). Defaults to toggled on.

Compress: Toggle on (default) to compress the payload body before sending.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests per Worker Process. When Cribl Edge hits this limit, it begins throttling traffic to the downstream service. Defaults to `5`. Minimum: `1`. Maximum: `32`. Each request can potentially hit a different HEC receiver.

Body size limit (KB): Maximum size, in KB, of the request body. Defaults to `5120`. Lowering the size can potentially result in more parallel requests and also cause outbound requests to be made sooner.

Events-per-request limit: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values can cause the payload size to be smaller than the configured **Body size limit**. Defaults to `5`.

- Retries happen on this flush interval.
- Any HTTP response code in the `2xx` range is considered success.
- Any response code in the `5xx` range is considered a retryable error, which will not trigger persistent queue (PQ) usage.
- Any other response code will trigger PQ (if PQ is configured as the Backpressure behavior).

Extra HTTP headers: Select **Add Header** to add **Name**/**Value** pairs to pass as additional HTTP headers. Values will be sent encrypted.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Cribl Edge will redact all headers, except non-sensitive headers that you declare below in **Safe headers**.

Safe headers: Add headers to declare them as safe to log in plaintext. (Sensitive headers such as `authorization` will always be redacted, even if listed here.) Use a tab or hard return to separate header names.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Notes on HTTP-Based Outputs

- To proxy outbound HTTP/S requests, see System Proxy Configuration.
- Cribl Edge will attempt to use keepalives to reuse a connection for multiple requests. After two minutes of the first use, the connection will be thrown away, and a new connection will be reattempted. This is to prevent sticking to a particular Destination when there is a constant flow of events.
- If the server does not support keepalives - or if the server closes a pooled connection while idle - a new connection will be established for the next request.

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

12.3.29. SentinelOne DataSet Destination

In a future release we're deprecating the SentinelOne DataSet Destination in favor of the SentinelOne AI SIEM Destination. The SentinelOne AI SIEM Destination uses SentinelOne's latest integration endpoints, offering enhanced flexibility and functionality.

Cribl Edge can send log events to the SentinelOne/Scalyr DataSet platform via the DataSet API. This Destination sends batches of events, as JSON, to that API's addEvents endpoint.

Type: Streaming | TLS Support: Yes | PQ Support: Yes

Configure Cribl Edge to Output to DataSet

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this output definition.
    - **Description**: Optionally, enter a description.
3. Under **Authentication**, select an **Authentication method** from the dropdown:
    - **Manual**: Displays a field for you to enter an **API key** that is available in your DataSet profile.
    - **Secret**: This option exposes an **API key (text secret)** drop-down to select a stored secret that references an API key. A **Create** link is available to store a new, reusable secret.
    - **API key**: Enter your DataSet API key that has `Log Write Access`.
4. Next, you can configure the following **Optional Settings**:
    - **DataSet site**: Select the US (default), `Europe`, or `Custom` region. If you select `Custom`, enter your custom endpoint URL.

- **Message field**: Name of the event field that contains the message to send. If not specified, Cribl Edge sends all non-internal fields of events passing through the Destination. For details, see Message Field.
- **Exclude fields**: Fields to exclude from the event if the **Message field** either is unspecified or refers to an object. Ignored if the **Message field** is a string, number, or boolean. If empty, Cribl Edge sends all non-internal fields. Default exclude fields are `sev`, `_time`, `ts`, and `thread`. We automatically send these fields as metadata of the event, in DataSet's required format. This is to avoid charges for field bytes - metadata bytes do not count toward ingestion.
- **Server/host field**: Name of the event field that contains the server or host that generated the event. Cribl Edge groups events by the value of this field, and gives them a unique session token to conform to the DataSet API. Each group is sent out as a separate batch; therefore, Cribl recommends specifying a field with a low cardinality, to avoid queuing up many different batches at the Destination. If not specified, or not a string, the implicit default value is `cribl_<outputId>`. For more details, see Rate Limiting and Session Management.
- **Timestamp field**: Name of the event field that contains the event timestamp. Cribl Edge sends this value as part of each event's metadata, not as an attribute field on the event.

  Timestamps are automatically converted to a nanosecond-precision string. If an event does not contain the field specified **Timestamp field**, or if the value cannot be converted into a nanosecond-precision string, Cribl Edge assigns a timestamp using the first valid output returned from `ts`, `_time`, or `Date.now()`, in that order.
- **Severity**: Use the drop-down to assign a default value to the `severity` field (which is sent as event metadata, not as an attribute field). Cribl Edge falls back to this value when an event contains no valid `sev` or `__severity` field. For details, see Severity.
- **Backpressure behavior**: Specify whether to block, drop, or queue events when all receivers are exerting backpressure. Defaults to `Block`.
- **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

5. Optionally, you can adjust the Persistent Queue, Processing, Retries and Advanced settings outlined in the sections below.
6. Select **Save**, then **Commit & Deploy**.

Message Field

If **Message field** is specified, we follow this logic:

- If an event does not contain the specified field, send the whole event (except internal fields).

- If an event has the specified field, and the field's value is a non-object, send the event in the format: `{ message: <value from event> }`.
- If an event has the specified field, and the field's value is an object, send the event in the format: `{ <all fields from the object> }`.

Severity

DataSet's severity model ranges from `0` least-severe (finest) to `6` most-severe (fatal).

- Where an event's `sev` field contains an integer in this range, Cribl Edge passes it through as the severity.
- Where the `sev` field contains a string matching DataSet's enum (`finest`, `finer`, `fine`, `info`, `warning`, `error`, `fatal`), Cribl Edge converts it to the corresponding integer.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

> On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.
>
> This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.

Mode	Description
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5 GB`. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.



Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Retries

Honor Retry-After header: Whether to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Edge limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. When enabled, any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. When disabled, all `Retry-After` headers are ignored.

Settings for failed HTTP requests: When you want to automatically retry requests that receive particular HTTP response status codes, use these settings to list those response codes.

For any HTTP response status codes that are not explicitly configured for retries, Cribl Edge applies the following rules:

Status Code	Action
Any in the `1xx`, `3xx`, or `4xx` series	Drop the request
Any in the `5xx` series	Retry the request

Upon receiving a response code that's on the list, Cribl Edge first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Edge continues retrying the request without increasing the time between retries any further.

If the sender (which manages the connection to the Destination) is at capacity, it will not accept any incoming events. These incoming events originate internally from a previous stage of the data flow when Destinations send outbound requests to their respective external services, and they include retry requests and new requests. Any events that were already in transit when the sender reached capacity will continue to be processed downstream.

Sender capacity is freed up when an outgoing request succeeds or encounters a non-retryable error. When the sender has available capacity again, it will resume accepting incoming events. This capacity management is influenced by the number of active connections and configured limits, such as concurrency and buffer sizes. If a Pipeline sends events faster than the Destination can process, the buffers may fill up, leading to backpressure and `Sender at capacity` warnings. This backpressure prevents the sender from accepting additional requests until capacity is restored.

By default, `429 (Too Many Requests)` is the only response code configured for automatic retries. For each response code you want to add to the list, select **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: When you want to automatically retry requests that have timed out, toggle this control on to display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).

- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Defaults to toggled on to reject certificates that are **not** authorized by a CA in the **CA certificate path**, nor by another trusted CA (for example, the system's CA).

Round-robin DNS: Toggle on to enable round-robin DNS lookup across multiple IP addresses, IPv4 and IPv6. When a DNS server resolves a Fully Qualified Domain Name (FQDN) to multiple IP addresses, Cribl Edge will sequentially use each address in the order they are returned by the DNS server for subsequent connection attempts.

Compress: Toggle on (default) to compress the payload body before sending.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests before blocking. This is set per Worker Process. Defaults to `5`.

Body size limit (KB): Maximum size of the request body before compression. Defaults to `4096` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Body size limit** value until downstream receivers reliably accept all events.

Buffer memory limit (KB): Total amount of memory used to buffer outgoing requests waiting to be sent. If left blank, defaults to 5 times the max body size (if set). If 0, no limit is enforced. This provides granular control over the memory allocated for buffering outgoing batched requests. Increasing the limit allows batches to grow larger before being flushed, improving efficiency for data with high cardinality (a large number of unique batches). Finding the optimal balance between efficient data transfer and memory usage involves adjusting both the **Buffer memory limit** and **Body size limit** settings.

Max events per request: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Extra HTTP headers: Click **Add header** to define **Name**/**Value** pairs to pass as additional HTTP headers.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Cribl Edge will redact all headers, except non-sensitive headers that you declare below in **Safe headers**.

Safe headers: Add headers here to declare them as safe to log in plaintext. (Sensitive headers like `authorization` will always be redacted, even if listed here.) Use a tab or hard return to separate header names.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Rate Limiting and Session Management

DataSet rate limits incoming data per session, not based on API key or IP address. The rate limit is 10 MB/sec per session, but the recommended limit is 2.5 MB/sec. To exceed this limit, split events into multiple sessions.

In Cribl Edge, the session contains:

- A unique session ID
- An optional `serverHost` describing the data source, defaulting to `cribl_${this.outId}`
- Optional properties like `logfile` and `parser` (not currently configurable)

Events are grouped into batches based on `serverHost`, with each batch having its own session ID and rate limit.

Configuration Tips

- **Default Configuration**: If **Server/host field** is left empty, all events will have `serverHost: cribl_${this.outId}`, sharing the same session ID and rate limit.
- **Increasing Sessions**: Set **Server/host field** to an event field with higher cardinality to create multiple sessions and increase the combined rate limit.
- **Practical Solutions**:
  - **Original Source Server/Host**: Set **Server/host field** to a field describing the original source server/host name.
  - **Worker Process ID**: In a Pipeline, set a field like `worker_id` to `C.env.CRIBL_WORKER_LABEL` and configure **Server/host field** to `worker_id`.
  - **Random Distribution**: Use an **Eval** function to set a field to `Math.floor(Math.random() * 8)` and configure **Server/host field** to this field to divide events into multiple sessions.

Internal Fields

The `__severity` field is included in the severity assignment order, after the `sev` field. The order is `sev`, `__severity`, then the configured default **Severity**.

Proxying Requests

If you need to proxy HTTP/S requests, see System Proxy Configuration.

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

12.3.30. ServiceNow Cloud Observability

The ServiceNow Cloud Observability Destination enables seamless integration with ServiceNow's Cloud Observability platform. This Destination routes observability data, including logs, metrics, and traces, to ServiceNow for enhanced monitoring and analysis. To ensure your data conforms to the expected format, refer to the Protocol section below.

To get started, connecting to ServiceNow Cloud Observability is simple. Grab your ServiceNow Cloud Observability access token, pick your preferred protocol and endpoint, then add the token to your Destination settings. Need more detailed instructions or configuration options? See below!



Type: Streaming | TLS Support: Configurable | PQ Support: Yes

Configuring a ServiceNow Cloud Observability Destination

In Cribl Edge, set up a ServiceNow Cloud Observability Destination.

1. From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:
    - To configure via QuickConnect, click **Routing** then **QuickConnect** (Cribl Stream) or **Collect** (Cribl Edge). Next, click **Add Destination** and from the resulting drawer's tiles, select **ServiceNow Cloud Observability**. Next, click either **Add Destination** or (if displayed) **Select Existing**.
    - To configure via the Routes, click **Data** then **Destinations** (Cribl Stream) or **More** then **Destinations** (Cribl Edge). Select **ServiceNow Cloud Observability** from the list of tiles or the **Destinations** list. Next, click **Add Destination** to open a **New Destination** modal.
2. In the Destination modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this output definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
    - **OTLP version**: Already set to `1.3.1` and cannot be changed.
    - **Protocol**: Use the drop-down to choose the protocol to use when sending data:
        - gRPC (default): Uses Protocol Buffers (Protobuf) for serialization.
        - HTTP: Uses Binary Protobuf encoding (does not support JSON Protobuf).
    - **Endpoint**: The endpoint to send data to ServiceNow. Default options are `ingest.lightstep.com:443` (United States) and `ingest.eu.lightstep.com:443` (Europe).
        - You can specify a custom endpoint using either a valid URL or an IP address (IPv4 or IPv6). Enclose IPv6 addresses in square brackets (for example,

`[2001:db8:85a3:0000:0000:8a2e:0370:7334]`).

- To proxy outbound HTTP/S requests, see System Proxy Configuration.

- **Auth token (text secret)**: Create or select a Cribl stored text secret that contains your ServiceNow Cloud Observability access token, ensuring secure authentication for data transmission.
- **Optional Settings**:
  - When using `HTTP`, you can customize the path instead of using the default cloud endpoints. This is useful when integrating with a hosted Lightstep satellite.
  - **Backpressure behavior**: Whether to `Block`, `Drop`, or `Persistent Queue` events when all receivers are exerting backpressure.
  - **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

3. Depending on the protocol you've selected, you have the following options:
   - **gRPC** provides TLS settings.
   - **HTTP** provides retry options, in addition to custom paths.
4. The **Processing Settings** and **Advanced Settings** provide granular control over how data is handled and transmitted.
   - Processing Settings allow you to define pre-processing and post-processing Pipelines, ensuring that events conform to the relevant Protobuf specifications for traces, metrics, and logs. This includes adding metadata, applying transformations, and conditioning the data.
   - Advanced Settings offer additional configuration options that enhance the flexibility and performance of data transmission.
     - For `gRPC`, this includes connection timeout, keep-alive time, and metadata headers.
     - For `HTTP`, this includes options like custom path overrides for traces, metrics, and logs, as well as settings for validating server certificates and using round-robin DNS. HTTP is simpler to set up and more widely compatible with existing tools and platforms.
5. Click **Save**, then **Commit & Deploy**.
6. Use the **Test** tab in the Destination's configuration modal to validate the configuration and connectivity of your Destination.

Custom HTTP Path

To send data to a custom location, override the default endpoint paths. When the **Protocol** is set to `HTTP`, this Destination sends data to the following endpoints by default:

- Traces: `<endpoint>/traces/otlp/v0.9`

- Metrics: `<endpoint>/metrics/otlp/v0.9`
- Logs: `<endpoint>/v1/logs`

`<endpoint>` refers to the base URL you specified in the **Endpoint** setting above.

To override a default path, provide only the desired path in the corresponding field: **Traces endpoint override**, **Metrics endpoint override**, or **Logs endpoint override**.

TLS Settings (Client Side)

TLS is available only when **General Settings** > **Protocol** is set to `gRPC`.

Use TLS Default is toggled off. When toggled on:

Validate server certs: Toggle on (default) to reject certificates that are **not** authorized by a CA in the **CA certificate path**, nor by another trusted CA (for example, the system's CA).

Minimum TLS version: Optionally, select the minimum TLS version to use when connecting.

Maximum TLS version: Optionally, select the maximum TLS version to use when connecting.

Certificate: Select an existing certificate name from the drop-down or add a new certificate by selecting **Create**. See Add a TLS Certificate to a Fleet for details.

CA certificate path: Path on client containing CA certificates (in PEM format) to use to verify the server's cert. Path can reference `$ENV_VARS`.

Private key path (mutual auth): Path on client containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Certificate path (mutual auth): Path on client containing certificates in (PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Passphrase: Passphrase to use to decrypt private key.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues

- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5 GB`. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.



Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output - both metric events, as dimensions; and log events, as labels. Supports wildcards.

By default, includes `cribl_pipe` (Cribl Edge Pipeline that processed the event).

Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.

- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Retries

This tab is displayed when **General Settings** > **Protocol** is set to `HTTP`.

Honor Retry-After header: Toggle on to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream/Edge limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. Any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. Toggle off to ignore all `Retry-After` headers.

Default Retry Behavior: By default, Cribl Stream/Edge applies predefined retry strategies for certain failed requests to the Destination using an exponential backoff algorithm. You can override these settings by manually configuring the retry behavior for a particular error code.

Destinations created before the 4.16.0 release do not support the default `4xx` series retry behavior. If your Destination was created prior to this release and you'd like it to support the behavior, reconfigure the Destination.

To retry requests, Cribl Stream/Edge applies the following rules by default:

Status Code	Action
Any in the `1xx`, `3xx` series	Drop the request
`401`, `403`, `408`, `429` errors	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached
Any other `4xx` series	Drop the request
Any in the `5xx` series	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached

By default, any `429 (Too Many Requests)` response that is not listed in the retry settings is treated as a non-retryable `4xx` error and dropped. If you want `429` responses retried, ensure `429` is included in the **Settings for failed HTTP requests**.

Settings for failed HTTP requests: You can also manually configure Cribl Stream/Edge to automatically retry requests that receive particular HTTP response status codes.

Upon receiving a response code that's on the list, Cribl Stream/Edge first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream/Edge continues retrying the request without increasing the time between retries any further.

If the sender (which manages the connection to the Destination) is at capacity, it will not accept any incoming events. These incoming events originate internally from a previous stage of the data flow when Destinations send outbound requests to their respective external services, and they include retry requests and new requests. Any events that were already in transit when the sender reached capacity will continue to be processed downstream.

Sender capacity is freed up when an outgoing request succeeds or encounters a non-retryable error. When the sender has available capacity again, it will resume accepting incoming events. This capacity management is influenced by the number of active connections and configured limits, such as concurrency and buffer sizes. If a Pipeline sends events faster than the Destination can process, the buffers may fill up, leading to backpressure and `Sender at capacity` warnings. This backpressure prevents the sender from accepting additional requests until capacity is restored.

For each response code you want to add to the list, select **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: Toggle on to automatically retry requests that have timed out and display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3

minutes).

Advanced Settings

This tab's options depend on whether **General Settings** > **Protocol** is set to the `gRPC` or `HTTP` transport.

Compression: Compression type to apply to messages sent to the OpenTelemetry endpoint. `Gzip` (the default) and `None` are available for both protocols; `Deflate` is available for gRPC only.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30` sec.

Request concurrency: Maximum number of concurrent requests before blocking. This is set per Worker Process. Defaults to `5`.

Body size limit (KB): Maximum size of the request body before compression. Defaults to `2048` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Body size limit** value until downstream receivers reliably accept all events.

Flush period (sec): Maximum time between requests. Low values could cause the payload size to be smaller than the configured **Max body size**. Defaults to `1` sec.

Auth token name: Name of the header or metadata key that will carry your ServiceNow Cloud Observability access token during data transmission. The default is `lightstep-access-token` and is customizable to match your specific requirements.

Environment: Optionally, specify a single Git branch on which to enable this configuration. If this field is empty, the config will be enabled everywhere.

Additional Settings for gRPC

When **General Settings** > **Protocol** is set to `gRPC`, the **Advanced Settings** tab adds the following options.

Connection timeout: Amount of time (milliseconds) to wait for the connection to establish before retrying. Defaults to `10000` (10 sec.).

Keep alive time (seconds): How often the sender should ping the peer to keep the connection alive. Defaults to `30`.

Metadata: Extra information to send with each gRPC request. Click **Add Metadata** to add each item as a **Key-Value** pair. The **Key** field is arbitrary. The **Value** field is a JavaScript expression that is evaluated just once, when this Destination is initialized. If you pass credentials as metadata, Cribl recommends using C.Secret().

Additional Settings for HTTP

When **General Settings** > **Protocol** is set to `HTTP`, the **Advanced Settings** tab adds the following options.

Validate server certs: Toggle on to reject certificates that are **not** authorized by a CA in the **CA certificate path**, nor by another trusted CA (e.g., the system's CA).

Round-robin DNS: Toggle on to enable round-robin DNS lookup across multiple IP addresses, IPv4 and IPv6. When a DNS server resolves a Fully Qualified Domain Name (FQDN) to multiple IP addresses, Cribl Edge will sequentially use each address in the order they are returned by the DNS server for subsequent connection attempts.

Keep alive: By default, Cribl Edge sends `Keep-Alive` headers to the remote server and preserves the connection from the client side up to a maximum of 120 seconds. Toggle this off if you want Cribl Edge to close the connection immediately after sending a request.

Extra HTTP headers: Click **Add Header** to define additional HTTP headers to pass to all events. Each row is a **Name-Value** pair. Values will be sent encrypted. You can also add headers dynamicall,y on a per-event basis, in the `__headers` field.

Safe headers: Add headers here to declare them as safe to log in plaintext. (Sensitive headers such as `authorization` will always be redacted, even if listed here.) Use a tab or hard return to separate header names.

Protocol and Transport Support

This Destination leverages the OpenTelemetry Protocol (OTLP) v1.3.1 to send observability data to ServiceNow. It supports both gRPC and HTTP protocols, ensuring flexibility in data transmission. OTLP defines Protocol buffer (Protobuf) schemas for its payloads (requests and responses). With the HTTP transport, this Destination supports Binary Protobuf payload encoding, but currently does not support JSON Protobuf.

When configuring Pipelines (including pre-processing and post-processing Pipelines), you need to ensure that events sent to this Destination conform to the relevant Protobuf specification.

- For traces, opentelemetry/proto/trace/v1/trace.proto
- For metrics, opentelemetry/proto/metrics/v1/metrics.proto
- For logs, opentelemetry/proto/logs/v1/logs.proto

This Destination will drop non-conforming events. If the Destination encounters a parsing error when trying to convert an event to OTLP, it discards the event and Cribl Edge logs the error.

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

12.3.31. SignalFx Destination

Cribl Edge supports sending events to SignalFx, now rebranded as Splunk Observability Cloud.



Type: Streaming | TLS Support: Yes | PQ Support: Yes

Configure Cribl Edge to Output to SignalFx

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this SignalFX definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
    - **Description**: Optionally, enter a description.
    - **Realm**: Observability Cloud realm name (for example, `us0`). Required.
3. In the **Authentication** settings, select one of the following options from the menu:
    - **Manual**: Displays an **Auth token** field for you to enter your Observability Cloud API access token. See the Authentication tokens topic in the API Reference.
    - **Secret**: This option exposes an **Auth token (text secret)** drop-down, in which you can select a stored secret that references the API access token described above. A **Create** link is available to store a new, reusable secret.
4. Next, you can configure the following **Optional Settings**:
    - **Backpressure behavior**: Select whether to block, drop, or queue events when all receivers are exerting backpressure. (Causes might include a broken or denied connection, or a rate limiter.) Defaults to `Block`.
    - **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

5. Optionally, you can adjust the Persistent Queue, Processing, Retries and Advanced settings outlined in the sections below.
6. Select **Save**, then **Commit & Deploy**.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.



If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5` GB. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.

Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Retries

Honor Retry-After header: Toggle on to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream/Edge limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. Any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. Toggle off to ignore all `Retry-After` headers.

Default Retry Behavior: By default, Cribl Stream/Edge applies predefined retry strategies for certain failed requests to the Destination using an exponential backoff algorithm. You can override these settings by manually configuring the retry behavior for a particular error code.

Destinations created before the 4.16.0 release do not support the default `4xx` series retry behavior. If your Destination was created prior to this release and you'd like it to support the behavior, reconfigure the Destination.

To retry requests, Cribl Stream/Edge applies the following rules by default:

Status Code	Action
Any in the `1xx`, `3xx` series	Drop the request

Status Code	Action
`401`, `403`, `408`, `429` errors	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached
Any other `4xx` series	Drop the request
Any in the `5xx` series	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached

By default, any `429 (Too Many Requests)` response that is not listed in the retry settings is treated as a non-retryable `4xx` error and dropped. If you want `429` responses retried, ensure `429` is included in the **Settings for failed HTTP requests**.

Settings for failed HTTP requests: You can also manually configure Cribl Stream/Edge to automatically retry requests that receive particular HTTP response status codes.

Upon receiving a response code that's on the list, Cribl Stream/Edge first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream/Edge continues retrying the request without increasing the time between retries any further.

If the sender (which manages the connection to the Destination) is at capacity, it will not accept any incoming events. These incoming events originate internally from a previous stage of the data flow when Destinations send outbound requests to their respective external services, and they include retry requests and new requests. Any events that were already in transit when the sender reached capacity will continue to be processed downstream.

Sender capacity is freed up when an outgoing request succeeds or encounters a non-retryable error. When the sender has available capacity again, it will resume accepting incoming events. This capacity management is influenced by the number of active connections and configured limits, such as concurrency and buffer sizes. If a Pipeline sends events faster than the Destination can process, the buffers may fill up, leading to backpressure and `Sender at capacity` warnings. This backpressure prevents the sender from accepting additional requests until capacity is restored.

For each response code you want to add to the list, select **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.

- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: Toggle on to automatically retry requests that have timed out and display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Toggle on to reject certificates that are **not** authorized by a CA in the **CA certificate path**, nor by another trusted CA - for example, the system's CA.

Round-robin DNS: Toggle on to enable round-robin DNS lookup across multiple IP addresses, IPv4 and IPv6. When a DNS server resolves a Fully Qualified Domain Name (FQDN) to multiple IP addresses, Cribl Edge will sequentially use each address in the order they are returned by the DNS server for subsequent connection attempts.

Compress: Toggle on (default) to compress the payload body before sending.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests per Worker Process. When Cribl Edge hits this limit, it begins throttling traffic to the downstream service. Defaults to `5`. Minimum: `1`. Maximum: `32`.

Body size limit (KB): Maximum size of the request body before compression. Defaults to `4096` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Body size limit** value until downstream receivers reliably accept all events.

Events-per-request limit: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values can cause the payload size to be smaller than the configured **Body size limit**. Defaults to `1` second.

Extra HTTP headers: Click **Add Header** to insert extra headers as **Name/Value** pairs. Values will be sent encrypted.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Cribl Edge will redact all headers, except non-sensitive headers that you declare below in **Safe headers**.

Safe headers: Add headers to declare them as safe to log in plaintext. (Sensitive headers such as `authorization` will always be redacted, even if listed here.) Use a tab or hard return to separate header names.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Proxying Requests

If you need to proxy HTTP/S requests, see System Proxy Configuration.

Notes About SignalFx

SignalFx has been rebranded as Splunk Observability Cloud, but the API endpoints still have URLs at `signalfx.com`.

For details on integrating with SignalFx, see the Developer Guide. Especially relevant is the Send Traces, Metrics and Events section of the Observability Cloud API Reference.

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

12.3.32. SNMP Trap Destination

Cribl Edge supports forwarding of SNMP Traps out.

Type: Streaming | TLS Support: No | PQ Support: No

If you've modified SNMP trap events within the Pipeline, use the SNMP Trap Serialize Function to serialize compliant events into SNMP traps before sending them to an SNMP Trap Destination.

Configure Cribl Edge to Forward to SNMP Traps

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this SNMP Trap definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
    - **Description**: Optionally, enter a description.
    - **SNMP trap destinations**: For each destination configured, enter the destination host **Address** and destination **Port**. **Port**: Defaults to `162`. To specify additional SNMP trap destinations to forward traps to on new rows, select **Add Destination**.
3. Next, you can configure the following **Optional Settings**:
    - **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
4. Optionally, you can adjust the Processing and Advanced settings outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

Advanced Settings

DNS resolution period (sec): Specify the interval (in seconds) to re-resolve hostnames. This reduces the frequency of DNS lookups, improving performance. Use this setting to balance the overhead of DNS lookup calls with the expected frequency of changes in the DNS records. Defaults to `0` seconds, meaning DNS lookups occur for every outgoing trap.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Considerations for Working with SNMP Traps Data

- It's possible to work with SNMP metadata (i.e., we'll decode the packet). Options include dropping, routing, etc. However, packets **cannot** be modified and sent to another SNMP Destination.
- SNMP packets can be forwarded to non-SNMP Destinations (for example, Splunk, Syslog, S3, etc.).
- SNMP packets can be forwarded to other SNMP Destinations. However, the contents of the incoming packet cannot be modified - i.e., we'll forward the packets verbatim as they came in.
- Non-SNMP input data **cannot** be sent to SNMP Destinations.

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

12.3.33. Sumo Logic Destination

Cribl Edge can send logs and metrics to Sumo Logic over HTTP. Sumo Logic offers a Hosted Collector that supports HTTP Sources to receive data over an HTTP POST request.

To send traces to Sumo Logic you can use Cribl's OpenTelemetry Destination pointing to Sumo Logic's OTLP/HTTP Source.



Type: Streaming | TLS Support: Configurable | PQ Support: Yes

How Sumo Logic Handles Data

- When an event contains the internal field `__criblMetrics`, it's sent to Sumo Logic as a metric event. Otherwise, it's sent as a log event.
- Data sent to Sumo Logic needs to be UTF-8 encoded and they recommend a data payload have a size, before compression, of 100 KB to 1 MB.
- Sumo Logic may impose throttling and caps on your log ingestion to prevent your account from using On-Demand Capacity, see Sumo Logic's manage ingestion documentation for details.

Prerequisites

In Sumo Logic, create or retrieve an HTTP Logs and Metrics Source's unique endpoint URL. See Sumo Logic's HTTP Logs and Metrics Source documentation for details. You will need the `Manage Collectors` role capability in Sumo Logic.

After creating the Source in Sumo Logic, the URL associated with the Source is displayed. Copy the endpoint URL so you can enter it when you configure the Cribl Edge Sumo Logic Destination.

Configure a Sumo Logic Destination

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.

- To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.

2. In the **New Destination** modal, configure the following under **General Settings**:
   - **Output ID**: Enter a unique name to identify this Sumo Logic Destination definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
   - **Description**: Optionally, enter a description.
   - **API URL**: Enter the endpoint URL of the Sumo Logic HTTP Source. This is provided by Sumo Logic after creating the Source, see prerequisites for details. For example, `https://endpoint6.collection.us2.sumologic.com/receiver/v1/http/<long-hash>`.
3. Next, you can configure the following **Optional Settings**:
   - **Custom source name** and **Custom source category** are unique settings to Sumo Logic. These allow you to override the Sumo Logic HTTP Source's **Source Name** and **Source Category** values with HTTP headers. Alternatively, you can define the `__sourceName` and `__sourceCategory` fields on events to assign a custom value at the event level.

   The remaining configurations are Cribl settings that you'll find across many Cribl Destinations.
   - **Backpressure behavior**: Whether to block, drop, or queue events when all receivers are exerting backpressure.
   - **Data Format**: This drop-down defaults to `JSON`. Change this to `Raw` if you prefer to preserve outbound events' raw format instead of JSONifying them.
   - **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
4. Optionally, configure any Persistent Queue, Processing, Retries, and Advanced settings outlined in the below sections.

> Requests to Sumo Logic are created in separate batches for each unique combination of **Source Name** and **Source Category** within the event payload. If you have a large number of unique pairs, this could lead to many individual batches being held in memory at any given time. You can increase the **Buffer memory limit** to give more memory allowance for buffering outgoing requests, allowing batches to grow larger before being flushed. The default flushing mechanism would discard smaller batches before they reach an expected size due to reaching the **Buffer memory limit**.

5. Click **Save**, then **Commit & Deploy**.
6. Verify that data is searchable in Sumo Logic. See the Verify Data Flow section below.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.



If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior.

Defaults to `5 GB`. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.



Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Retries

Honor Retry-After header: Toggle on to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream/Edge limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. Any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. Toggle off to ignore all `Retry-After` headers.

Default Retry Behavior: By default, Cribl Stream/Edge applies predefined retry strategies for certain failed requests to the Destination using an exponential backoff algorithm. You can override these settings by manually configuring the retry behavior for a particular error code.



Destinations created before the 4.16.0 release do not support the default `4xx` series retry behavior. If your Destination was created prior to this release and you'd like it to support the behavior, reconfigure the Destination.

To retry requests, Cribl Stream/Edge applies the following rules by default:

Status Code	Action
Any in the `1xx`, `3xx` series	Drop the request
`401`, `403`, `408`, `429` errors	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached
Any other `4xx` series	Drop the request
Any in the `5xx` series	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached

By default, any `429 (Too Many Requests)` response that is not listed in the retry settings is treated as a non-retryable `4xx` error and dropped. If you want `429` responses retried, ensure `429` is included in the **Settings for failed HTTP requests**.

Settings for failed HTTP requests: You can also manually configure Cribl Stream/Edge to automatically retry requests that receive particular HTTP response status codes.

Upon receiving a response code that's on the list, Cribl Stream/Edge first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream/Edge continues retrying the request without increasing the time between retries any further.

If the sender (which manages the connection to the Destination) is at capacity, it will not accept any incoming events. These incoming events originate internally from a previous stage of the data flow when Destinations send outbound requests to their respective external services, and they include retry requests and new requests. Any events that were already in transit when the sender reached capacity will continue to be processed downstream.

Sender capacity is freed up when an outgoing request succeeds or encounters a non-retryable error. When the sender has available capacity again, it will resume accepting incoming events. This capacity management is influenced by the number of active connections and configured limits, such as concurrency and buffer sizes. If a Pipeline sends events faster than the Destination can process, the buffers may fill up, leading to backpressure and `Sender at capacity` warnings. This backpressure prevents the sender from accepting additional requests until capacity is restored.

For each response code you want to add to the list, select **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: Toggle on to automatically retry requests that have timed out and display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).

- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Toggle on to reject certificates that are **not** authorized by a CA in the **CA certificate path**, nor by another trusted CA (for example, the system's CA).

Round-robin DNS: Toggle on to enable round-robin DNS lookup across multiple IP addresses, IPv4 and IPv6. When a DNS server resolves a Fully Qualified Domain Name (FQDN) to multiple IP addresses, Cribl Edge will sequentially use each address in the order they are returned by the DNS server for subsequent connection attempts.

Compress: Toggle on (default) to compress the payload body before sending.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests per Worker Process. When Cribl Edge hits this limit, it begins throttling traffic to the downstream service. Defaults to `5`. Minimum: `1`. Maximum: `32`.

Body size limit (KB): Maximum size of the request body before compression. Defaults to `1024` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Body size limit** value until downstream receivers reliably accept all events.

Buffer memory limit (KB): Total amount of memory used to buffer outgoing requests waiting to be sent. If left blank, defaults to 5 times the max body size (if set). If 0, no limit is enforced. This provides granular control over the memory allocated for buffering outgoing batched requests. Increasing the limit allows batches to grow larger before being flushed, improving efficiency for data with high cardinality (a large number of unique batches). Finding the optimal balance between efficient data transfer and memory usage involves adjusting both the **Buffer memory limit** and **Body size limit** settings.

Events-per-request limit: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Extra HTTP headers: Name-value pairs to pass as additional HTTP headers. Values will be sent encrypted.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Cribl Edge will redact all headers, except non-sensitive headers that you declare below in **Safe headers**.

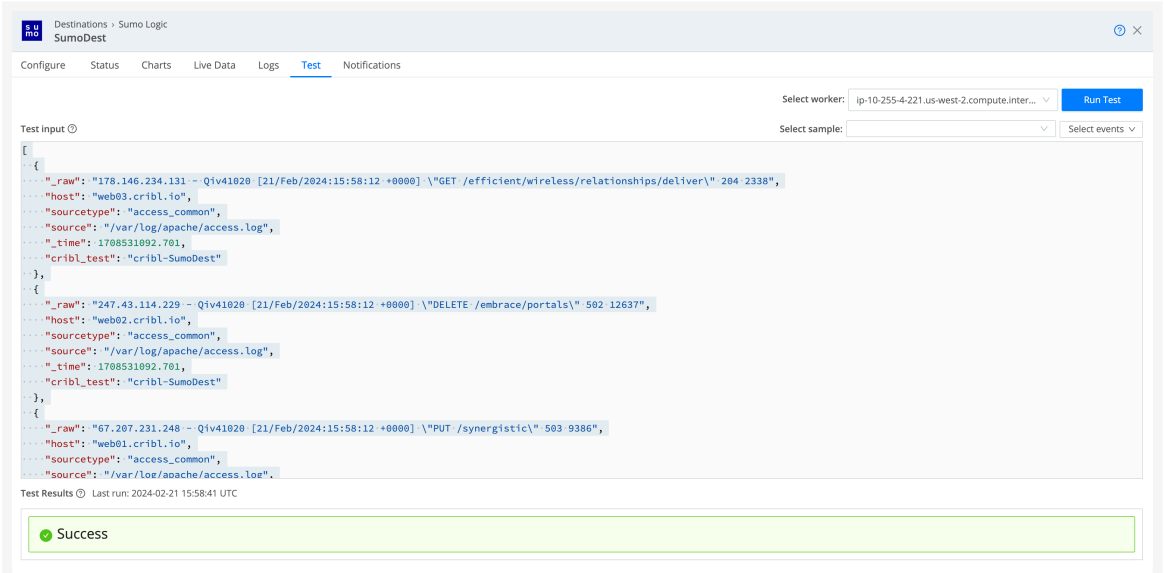
Safe headers: Add headers to declare them as safe to log in plaintext. (Sensitive headers such as `authorization` will always be redacted, even if listed here.) Use a tab or hard return to separate header names.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Verify Data Flow

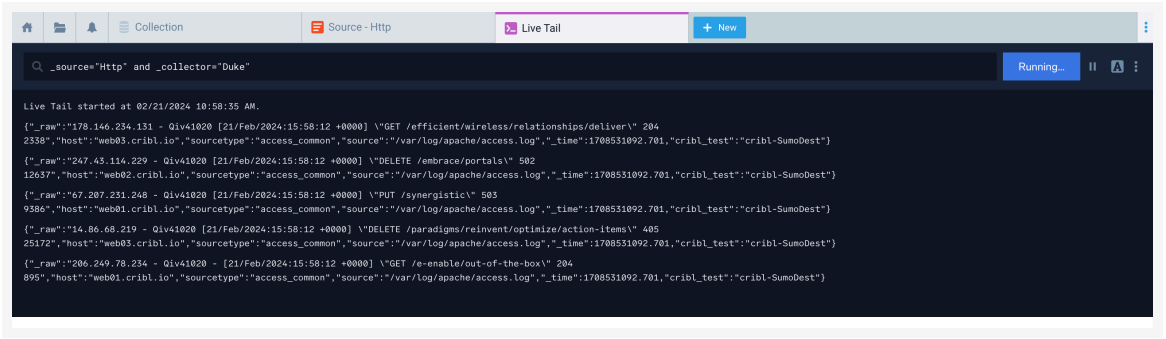
To verify data flow, we'll use the Destination's **Test** feature while running a **Live Tail** session in Sumo Logic.

1. In Sumo Logic, run a new Live Tail session against the name of the HTTP Logs and Metrics Source or the **Custom source name** you defined when you configured the Sumo Logic Destination. You'll need to use the `_source` metadata field. For example, if the name of the Source is `HTTP` your search would be `_source="HTTP"`. Ensure you've clicked the **Run** button to start the Live Tail session.
2. In Cribl Edge, open the Destination configuration modal and select the **Test** tab. You can leave the default data or select from the available samples. Click **Run Test**.



Example Destination Test

3. Look back to your Sumo Logic Live Tail session and you should see the sample data from the Cribl test displayed in your Live Tail results.



Example Live Tail Results

Troubleshooting

If you receive an error when verifying data flow, take note of the response code and reference Sumo Logic's documentation for common issues to investigate.

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

Notes on HTTP-Based Outputs

- To proxy outbound HTTP/S requests, see System Proxy Configuration.
- Cribl Edge will attempt to use keepalives to reuse a connection for multiple requests. After two minutes of the first use, the connection will be thrown away, and a new one will be reattempted. This is to prevent sticking to a particular destination when there is a constant flow of events.
- If the server does not support keepalives (or if the server closes a pooled connection while idle), a new connection will be established for the next request.
- When resolving the Destination's hostname, Cribl Edge will pick the first IP in the list for use in the next connection. Enable Round-robin DNS to better balance distribution of events between destination cluster

12.3.34. Syslog Destination

Cribl Edge supports sending out data over syslog via TCP or UDP.

Type: Streaming | TLS Support: Configurable | PQ Support: Yes

This Syslog Destination supports RFC 3164 and RFC 5424. **Before** you configure this Destination, review Syslog Format Options and Structure Syslog Output below. This will help you ensure that your configuration will structure RFC-compliant outbound events that downstream services can read. Additionally, the Syslog Destination does not compress outbound data. Any compression settings you configure under Persistent Queue affect only how queued data is stored on disk.

Configure Cribl Edge to Output Data in Syslog Format

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this Syslog definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
    - **Description**: Optionally, enter a description.
    - **Protocol**: The network protocol to use for sending out syslog messages. Defaults to `TCP`; `UDP` is also available.
    - **Load balancing**: Cribl Edge displays this option only when the **Protocol** is set to `TCP`. Toggle on to specify multiple destinations. See Load Balancing Settings below.

    Turning **Load balancing** off displays the following two additional fields:
    - **Address**: Address/hostname of the receiver.
    - **Port**: Port number to connect to on the host.

    If **Load balancing** is disabled and you notice that Cribl Edge is not sending data to all possible IP

addresses, enable **Advanced Settings** > **Round-robin DNS**.

- **Record size limit**: Displayed when **Protocol** is set to `UDP`. Sets the maximum size of syslog messages. Defaults to `1500`, and must be ≤ `2048`. To avoid packet fragmentation, keep this value less than or equal to the MTU (maximum transmission unit) for the network path to the downstream receiver. When the message length exceeds the size limit, it is truncated and followed with an ellipsis ("…").

3. Next, you can configure the following **Optional Settings**:
   - **Facility**: Default value for message facility. If set, will be overwritten by the value of `__facility`. Defaults to `user`.
   - **Severity**: Default value for message severity. If set, will be overwritten by the value of `__severity`. Defaults to `notice`.
   - **App name**: Default value for application name. If set, will be overwritten by the value of `__appname`. Defaults to `Cribl`.
   - **Message format**: The syslog message format supported by the receiver. Defaults to `RFC3164`.
   - **Timestamp format**: The timestamp format to use when serializing an event's time field. Options include `Syslog` (default) and `ISO8601`. For details, see Timestamp format below.
   - **Throttling**: Throttle rate, in bytes per second. Defaults to `0`, meaning no throttling. Multiple-byte units such as KB, MB, or GB are also allowed - for example, `42 MB`. When throttling is engaged, your **Backpressure behavior** selection determines whether Cribl Edge will handle excess data by blocking it, dropping it, or queueing it to disk. For details, see Throttling below.
   - **Backpressure behavior**: Select whether to block, drop, or queue events when all receivers are exerting backpressure. (Causes might include a broken or denied connection, or a rate limiter.) Defaults to `Block`.
   - **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
4. Optionally, you can adjust the Persistent Queue, TLS, Processing, Timeout, and Advanced settings outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

Load Balancing Settings



Load balancing is available only when the **Protocol** is set to `TCP`.

Toggling **Load balancing** on replaces the static **General Settings** > **Address** and **Port** fields with the following controls:

Exclude current host IPs: Appears when **Load balancing** is toggled on. It determines whether to exclude all IPs of the current host from the list of any resolved hostnames. Default is toggled off, which keeps the current host available for load balancing.

Destinations

Use the **Destinations** table to specify a known set of receivers on which to load-balance data. To specify more receivers on new rows, click **Add Destination**. Each row provides the following fields:

Address: Hostname of the receiver. Optionally, you can paste in a comma-separated list, in `<host>:<port>` format.

Port: Port number to send data to on this host.

TLS: Whether to inherit TLS configs from group setting, or disable TLS. Defaults to `inherit`.

TLS servername: Servername to use if establishing a TLS connection. If not specified, defaults to connection host (if not an IP). Otherwise, uses the global TLS settings.

Load weight: Set the relative traffic-handling capability for each connection by assigning a weight (> `0`). This column accepts arbitrary values, but for best results, assign weights in the same order of magnitude to all connections. Cribl Edge will attempt to distribute traffic to the connections according to their relative weights.

The final column provides an `X` button to delete any row from the table.

For details on configuring all these options, see About Load Balancing.

Timestamp Format

For `Syslog`, the timestamp format is `Mmm dd hh:mm:ss`, using the timezone of the syslog device. For example: `Jan 18 16:56:36`.

For `ISO8601`, the basic timestamp format is `YYYY-MM-DD`. For example: `2023-01-18`. For extended formats, add hours, minutes, seconds, decimal fractions (optional), and timezone. For example: `2023-01-18T16:56:36.996-08:00`.

Throttling

When **Throttling** is used in conjunction with **Connection limit**, the throttling rate is applied per connection. For instance, if **Connection limit** is set to `2` and throttling is set to `3 MB`, each of the two connections can send data at a rate of up to 3 MB per second, resulting in a total potential throughput of 6 MB per second across both connections.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.



If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5` GB. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.



Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

TLS Settings (Client Side)

Enabled: Defaults to toggled off. When toggled on:

Validate server certs: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA - for example, the system's CA. Defaults to toggled on.

Server name (SNI): Server name for the SNI (Server Name Indication) TLS extension. This must be a host name, not an IP address.

Minimum TLS version: Optionally, select the minimum TLS version to use when connecting.

Maximum TLS version: Optionally, select the maximum TLS version to use when connecting.

Certificate: Select an existing certificate name from the drop-down or add a new certificate by selecting **Create**. See Add a TLS Certificate to a Fleet for details.

CA certificate path: Path on client containing CA certificates (in PEM format) to use to verify the server's cert. Path can reference `$ENV_VARS`.

Private key path (mutual auth): Path on client containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Certificate path (mutual auth): Path on client containing certificates in (PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Passphrase: Passphrase to use to decrypt private key.

Timeout Settings



These timeout settings apply only to the TCP protocol.

Connection timeout: Amount of time (in milliseconds) to wait for the connection to establish, before retrying. Defaults to `10000`.

Write timeout: Amount of time (milliseconds) to wait for a write to complete, before assuming connection is dead. Defaults to `60000`.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Advanced Settings

The first three options are always displayed:

Octet count framing: When enabled, Cribl Edge prefixes messages with their byte count, regardless of whether the messages are constructed from `message` or `__syslogout`. When disabled, Cribl Edge omits prefixes, and instead appends a `\n` to messages. RFCs 5425 and 6587 describe how octet counting works in syslog.

Enable Source IP spoofing: Cribl Edge displays this option only when **Protocol** is set to `UDP`. This feature is available only for on-prem or hybrid Worker Groups (not available in Cribl.Cloud or on Cribl Edge Nodes). Toggle on to use the event Source IP and port for outgoing UDP packets.

This field preserves the event's original IP address and port from the internal `__srcIpPort` field, rather than the Worker Process's IP. This is useful when sending data to systems that rely on the Source IP to identify the original sender. For the prerequisites, see `udp-sender` Helper Installation and Setup.

Log failed requests to disk: Toggle on to make the payload of the most recently failed request available for inspection. See Inspect Payload to Troubleshoot Closed Connections below.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

The following options are added if you enable the General Settings tab's **Load balancing** option:

DNS resolution period (seconds): Re-resolve any hostnames after each interval of this many seconds, and pick up destinations from A records.

- For TCP: This helps maintain up-to-date IP addresses for your TCP connections. Defaults to `600` seconds.
- For UDP: This reduces the frequency of DNS lookups for UDP destinations, improving performance. Defaults to `0` seconds, meaning DNS lookups occur for every outgoing message.

Load balance stats period (seconds): Lookback traffic history period. Defaults to `300` seconds. If multiple receivers - that is, multiple A records - are behind a hostname, all resolved IPs will inherit the weight of the host

unless each IP is specified separately. In Cribl Edge load balancing, IP settings take priority over those from hostnames.

Connection limit: Constrains the number of concurrent receiver connections per Worker Process, to limit memory utilization. If you set this to a number greater than `0`, Cribl Edge will randomly select that many discovered IPs to connect to on every DNS resolution period. Cribl Edge will rotate IPs in future resolution periods - monitoring weight and historical data, to ensure fair load balancing of events among IPs.

`udp-sender` Helper Installation and Setup

Enabling IP spoofing requires creating raw network sockets, which demands elevated system privileges. To maintain the Cribl Stream security model, this operation is isolated into a dedicated helper program. IP spoofing is not available on Cribl Edge.

To enable IP spoofing, complete the following installation and privilege configuration steps on every Worker Node:

1. Download the `udp-sender` helper binary. The source code and releases are available on the public GitHub repository: https://github.com/criblio/udp-sender/.
2. Install the executable binary at the required path: `/usr/bin/udp-sender`
3. Set permissions: A system administrator with root access must grant the binary the necessary Linux capability (`CAP_NET_RAW`). This allows the helper to construct and send raw network packets. Use the following command to set the capability:
   ```
   sudo setcap 'cap_net_raw+eip' /usr/bin/udp-sender
   ```
4. If you are using package installers, the installation script will automatically create a custom `udp-senders` group and set the necessary capabilities on the binary using the `setcap` command. You will need to manually ensure the `cribl` user is added to this group.



If the `udp-sender` helper fails to start (for example, due to missing permissions), the Destination will enter into an error state. Check the Worker Process logs for an error message indicating a failure to create a raw socket, which confirms the `CAP_NET_RAW` capability has not been correctly applied to the `/usr/bin/udp-sender` file.

For additional technical details on the binary's function and installation, consult the README in the public GitHub repository.

Inspect Payload to Troubleshoot Closed Connections

When a downstream receiver closes connections from this Destination (or just stops responding), inspecting the payload of the most recently failed request can help you find the cause. For example:

- Suppose you send an event whose size is larger than the downstream receiver can handle.
- Suppose you send an event that has a `number` field, but the value exceeds the highest number that the downstream receiver can handle.

When **Log failed requests to disk** is enabled, you can inspect the last failed request payload. Here is how:

1. In the Destination UI, navigate to the **Logs** tab.
2. Find a log entry with a `connection error` message.
3. Expand the log entry.
4. If the message includes the phrase `See payload file for more info`, note the path in the `file` field on the next line.

Now you have the path to the directory where Cribl Edge is storing the payload from the last failed request.

Internal Fields

Cribl Edge uses a set of internal fields to assist in forwarding data to downstream services. Fields for this Destination:

- `__priority`
- `__facility`
- `__severity`
- `__procid`
- `__appname`
- `__msgid`
- `__syslogout`

Syslog Format Options

Unlike other Cribl Edge Destinations, the Syslog Destination applies an additional post-processing step after the Pipeline(s) transform events. This additional step structures the data for compliance with the syslog transport protocol (RFC 3164 and/or RFC 5424) before it is transmitted to downstream services.

- With RFC 3164-compliant messages, you'll get `priority`, `timestamp`, `host`, and `message`, such that `message` includes `appname` and `procid`. You will not get `msgid` or `STRUCTURED-DATA`.
- With RFC 5424-compliant messages - that is, structured data - Cribl Edge will send the contents of `event.structuredData`, whether or not that content is valid according to the RFC. Cribl Edge will make no attempt to frame, enclose, or validate the data. If present, structured data **should** be in the form `[`

`SD-ID *PARAM-NAME=PARAM-VALUE]` but Cribl Edge will not enforce this. Cribl Edge will send whatever is in that field, or, if the field is missing, send a dash (`-`).

In **General Settings**, you can format the timestamps, format the message delivering the event, and set defaults for **Facility**, **Severity**, and **App name** (in RFC 5424, these are Facility, Severity, and APP-NAME, respectively).

Here is an example of an RFC 3164-compliant syslog event:

- `<13>Jul 11 10:34:35 testbox testing[42]`

Here is an example of an RFC 5424-compliant syslog event:

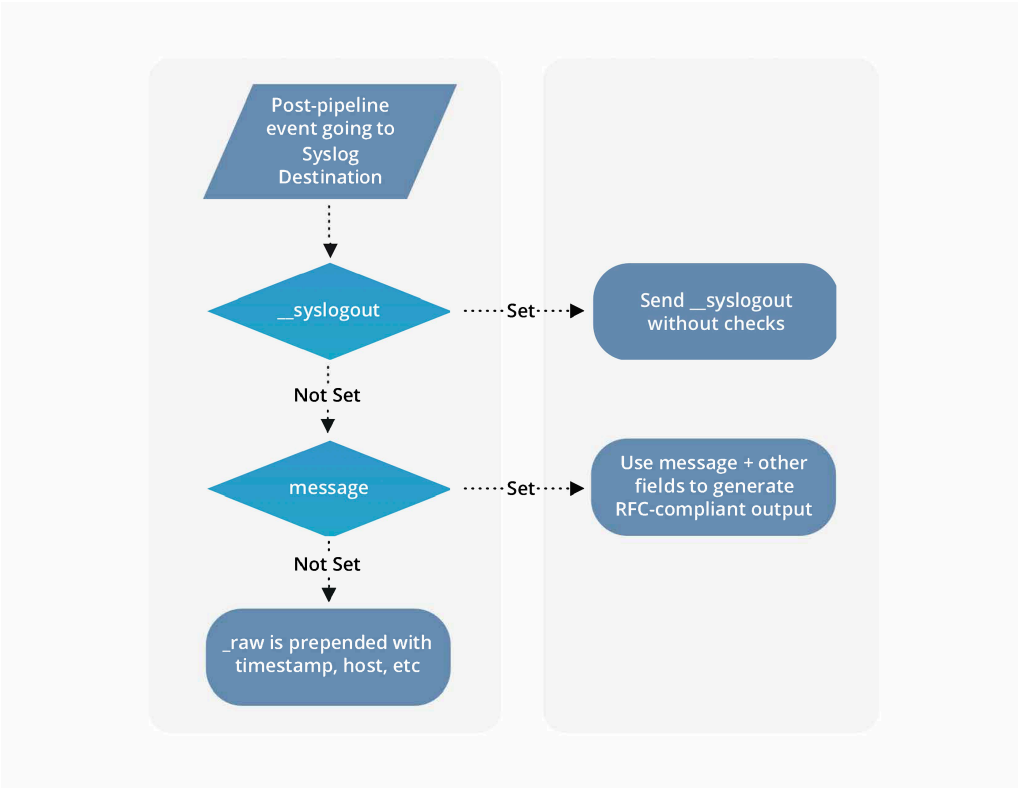
- `<214>1 2022-07-11T18:58:45.000Z testbox testapp 42 - [exampleSDID@32473 iut="3" foo="bar" this="that"] Here is a message`

Structure Syslog Output

Two approaches are available for specifying how the Syslog Destination will structure its output. One relies on the `message` field, and the other relies on the `__syslogout` field. The Syslog Destination determines what to do based on whether or not those fields are set, as the flowchart below makes clear.

- **Use `message` (Simplest Approach)**: Cribl recommends using this option, which requires you to configure `message` and to leave `__syslogout` unset. When `message` is defined, Cribl Edge discards `_raw` and composes a new payload using the other syslog-related fields. Cribl Edge automatically processes the values of the `message`, `_time`, `host`, and other fields, and creates an ISO-compliant timestamp and a properly formatted event. In the flowchart, the rounded rectangle that reads "Use message + other fields ..." represents this approach.
- Export `__syslogout` (Alternative Approach): When you define `__syslogout`, Cribl Edge sends this field and its value as the entire syslog message, discards `_raw`, and ignores all the other fields. In the flowchart, the rounded rectangle that reads "Send __syslogout ..." represents this approach.

As the flowchart shows, if **neither** `message` nor `__syslogout` are set, Cribl Edge uses the existing `_raw` field as the `message` field (with or without a header), and prepends all the other syslog-related fields to the event. While this could be considered a third approach, Cribl does not recommend it, and we will not describe it further in this topic.



Syslog output flow

Before you configure this Destination, review the subsection below that explains the approach that you prefer.

Use `message` (Simplest Approach)

Cribl recommends this approach for environments where events enter Cribl Edge from the Syslog Source, because that Source supplies all required fields.

This approach is the simplest and least error-prone because Cribl Edge creates the payload for you. The following fields are required:

- `facility`
- `severity`
- `_time`
- `host`
- `message`

Cribl Edge will create a new syslog-adherent payload using these required fields. See Field Logic and Behavior below. To specify which RFC your events should adhere to, use **General Settings** > **Optional Settings** > **Message format**.

If these optional fields are included in the event, Cribl Edge will use them appropriately when constructing the payload:

- `appname`
- `host`
- `procid`
- `msgid` (RFC 5424 only)
- `structuredData` (RFC 5424 only)

Field Logic and Behavior

Let's examine the fields in a syslog-formatted event. Keep in mind that if Severity, Facility, or APP-NAME are not set on the event, Cribl Edge defaults to the corresponding value(s) configured in the Syslog Destination (**Severity**, **Facility**, and **App name**, respectively).

- `__priority`: If this field is present Cribl Edge will use it and ignore `facility` and `severity`.
- The version (used for RFC 5424 only, and not a field that you can set) follows priority (with no spaces in between).
- `__facility` or `facility`: Cribl Edge uses this field to calculate priority. The RFCs define Facility codes.
- `__severity` or `severity`: Cribl Edge also uses this field to calculate priority. The RFCs define Severity levels.

Use either (a) `__facility` and `__severity` or (b) `facility` and `severity`. For these fields, avoid mixing double-underscore naming with no-underscore naming, because if you mix them, Cribl Edge will ignore one of them and you will get unintended results.



If `__priority` is not set, Cribl Edge calculates the Priority from the values of Severity and Facility. Cribl Edge uses the formula: Priority = ((8 * Facility) + Severity).

- For example, if the `facility` is `13` (Security) and the `severity` is `2` (Critical), the `priority` will be `(13 * 8) + 2 = 106`.

Now let's look at the rest of the fields. The list below shows field names in priority order: if both `__appname` and `appname` are present, Cribl Edge uses `__appname`.

- `_time`: The value of `_time` in Cribl events is in epoch format, but the syslog RFCs dictate that each event's timestamp must be in human-readable format (see TIMESTAMP in RFC 3164 or RFC 5424). When defining a Syslog Destination, you configure this with **General Settings** > **Timestamp format**. The `ISO8601` option (ISO 8601 format) defines the method for specifying time zone and year, while the older `Syslog` format lacks this information. For this reason, Cribl recommends `ISO8601`.

- `host`: the value for the required host field in a syslog event (see HOSTNAME in RFC 3164 or RFC 5424), following the timestamp.
- `__appname` or `appname`: The required application name (APP-NAME in RFC 5424). This is typically the name of the daemon or process that is logging any given event.
- `__procid` or `procid`: This optional field (PROCID in RFC 5424) is the Process ID. Use a numeric value, optionally surrounded with brackets - for example, `[4321]`. Cribl Edge will automatically adjust the spaces and syntax to ensure RFC-compliant formatting.
- `__msgid` or `msgid`: This optional field (MSGID in RFC 5424 only) is the Message ID.
- `structuredData`: This optional field (STRUCTURED-DATA in RFC 5424 only) contains a namespace and zero or more key-value pairs. Cribl Edge does not validate whether it adheres to the RFC.

Export `__syslogout` (Alternative Approach)



The most common use of this approach is to send to a non-syslog downstream receiver such as a raw TCP listener. While there is no native "raw TCP" Destination in Cribl Edge, this approach - where you configure a Syslog Destination with the TCP protocol and `__syslogout` - can be an effective method for delivering raw data over TCP to a downstream receiver that does not enforce syslog RFCs.

When you set `__syslogout`, that field becomes the entire syslog message sent. Neither `_raw` nor any other metadata are sent downstream.

If you enable **Octet count framing**, Cribl Edge will prepend the number of bytes of the constructed `__syslogout` field to the message before sending it.

Cribl Edge does not check whether the value of `__syslogout` is RFC-compliant. This means that to send proper, RFC-compliant syslog messages, you must manually construct the `__syslogout` payload, starting with `_time`, for all the fields that Cribl Edge would automatically handle with the first approach above.



If the value of `__syslogout` fails to adhere to the syslog RFCs, a downstream syslog server may misinterpret the event; drop non-compliant events entirely; or, try to "fix" the format by supplying missing fields.

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

12.3.35. TCP JSON Destination

Cribl Edge supports sending data over TCP in JSON format.



Type: Streaming | TLS Support: Configurable | PQ Support: Yes

Configure Cribl Edge to Output Data in TCP JSON Format

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this TCP JSON definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
    - **Description**: Optionally, enter a description.
    - **Load balancing**: Toggle on to specify multiple destinations. For details, see Load Balancing Settings below.
      The following two fields appear **only** when toggled off.
    - **Address**: Hostname of the receiver.
    - **Port**: Port number to connect to on the host.
3. In the **Authentication** settings, select one of the following options from the menu:
    - **Manual**: In the resulting **Auth token** field, you can optionally enter an auth token to use in the connection header.
    - **Secret**: This option exposes an **Auth token (text secret)** drop-down, in which you can select a stored secret that references the `authToken` header field value described above. A **Create** link is available to store a new, reusable secret.
4. Next, you can configure the following **Optional Settings**:
    - **Exclude current host IPs**: Appears when **Load balancing** is toggled on. It determines whether to exclude all IPs of the current host from the list of any resolved hostnames. Default is toggled off,

which keeps the current host available for load balancing.

- **Compression**: Codec to use to compress the data before sending. Defaults to `Gzip`.
- **Throttling**: Throttle rate, in bytes per second. Defaults to `0`, meaning no throttling. Multiple-byte units such as KB, MB, GB, and so forth, are also allowed - for example, `42 MB`. When throttling is engaged, your **Backpressure behavior** selection determines whether Cribl Edge will handle excess data by blocking it, dropping it, or queueing it to disk. For details, see Throttling below.
- **Backpressure behavior**: Specifies whether to block, drop, or queue events when all receivers are exerting backpressure. Defaults to `Block`. For details, see Persistent Queue Settings below.
- **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

5. Optionally, you can adjust the Persistent Queue, TLS, Processing, and Advanced settings outlined in the sections below.
6. Select **Save**, then **Commit & Deploy**.

Load Balancing Settings

Enabling the **Load balancing** toggle replaces the static **General Settings** > **Address** and **Port** fields with the following controls:

Destinations

Use the **Destinations** table to specify a known set of receivers on which to load-balance data. To specify more receivers on new rows, click **Add Destination**. Each row provides the following fields:

Address: Hostname of the receiver. Optionally, you can paste in a comma-separated list, in `<host>:<port>` format.

Port: Port number to send data to on this host.

TLS: Whether to inherit TLS configs from group setting, or disable TLS. Defaults to `inherit`.

TLS servername: Servername to use if establishing a TLS connection. If not specified, defaults to connection host (if not an IP). Otherwise, uses the global TLS settings.

Load weight: Set the relative traffic-handling capability for each connection by assigning a weight (> `0`). This column accepts arbitrary values, but for best results, assign weights in the same order of magnitude to all connections. Cribl Edge will attempt to distribute traffic to the connections according to their relative weights.

The final column provides an `X` button to delete any row from the table.

For details on configuring all these options, see About Load Balancing.

Throttling

When used **Throttling** is used in conjunction with **Connection limit**, the throttling rate is applied per connection. For instance, if **Connection limit** is set to `2` and throttling is set to `3 MB`, each of the two connections can send data at a rate of up to 3 MB per second, resulting in a total potential throughput of 6 MB per second across both connections.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

> On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.
>
> This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5 GB`. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.



Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

TLS Settings (Client Side)

Use TLS: Defaults to toggled off. When toggled on:

Autofill?: This setting is experimental.

Validate server certs: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (for example, the system's CA). Defaults to toggled on.

Server name (SNI): Server name for the SNI (Server Name Indication) TLS extension. This must be a host name, not an IP address.

Minimum TLS version: Optionally, select the minimum TLS version to use when connecting.

Maximum TLS version: Optionally, select the maximum TLS version to use when connecting.

Certificate: Select an existing certificate name from the drop-down or add a new certificate by selecting **Create**. See Add a TLS Certificate to a Fleet for details.

CA certificate path: Path on client containing CA certificates (in PEM format) to use to verify the server's cert. Path can reference `$ENV_VARS`.

Private key path (mutual auth): Path on client containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Certificate path (mutual auth): Path on client containing certificates in (PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Passphrase: Passphrase to use to decrypt private key.

Timeout Settings

Connection timeout: Amount of time (in milliseconds) to wait for the connection to establish before retrying. Defaults to `10000`.

Write timeout: Amount of time (in milliseconds) to wait for a write to complete before assuming connection is dead. Defaults to `60000`.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Advanced Settings

Log failed requests to disk: Toggle on to make the payload of the most recently failed request available for inspection. See Inspect Payload to Troubleshoot Closed Connections below.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

In General Settings, toggle **Load balancing** on to add the following settings:

DNS resolution period (seconds): Re-resolve any hostnames after each interval of this many seconds, and pick up destinations from A records. Defaults to `600` seconds.

Load balance stats period (seconds): Lookback traffic history period. Defaults to `300` seconds. (Note that If multiple receivers are behind a hostname - i.e., multiple A records - all resolved IPs will inherit the weight of the host, unless each IP is specified separately. In Cribl Edge load balancing, IP settings take priority over those from hostnames.)

Connection limit: Constrains the number of concurrent receiver connections, per Worker Process, to limit memory utilization. If set to a number > `0`, then on every DNS resolution period, Cribl Edge will randomly select this subset of discovered IPs to connect to. Cribl Edge will rotate IPs in future resolution periods - monitoring weight and historical data, to ensure fair load balancing of events among IPs.

Inspect Payload to Troubleshoot Closed Connections

When a downstream receiver closes connections from this Destination (or just stops responding), inspecting the payload of the most recently failed request can help you find the cause. For example:

- Suppose you send an event whose size is larger than the downstream receiver can handle.
- Suppose you send an event that has a `number` field, but the value exceeds the highest number that the downstream receiver can handle.

When **Log failed requests to disk** is enabled, you can inspect the last failed request payload. Here is how:

1. In the Destination UI, navigate to the **Logs** tab.
2. Find a log entry with a `connection error` message.
3. Expand the log entry.
4. If the message includes the phrase `See payload file for more info`, note the path in the `file` field on the next line.

Now you have the path to the directory where Cribl Edge is storing the payload from the last failed request.

Format

TCP JSON events are sent in newline-delimited JSON format, consisting of:

1. A header line. Can be empty, for example: `{}`. If **Auth Token** is enabled, the token will be included here as a field called `authToken`. In addition, if events contain common fields, they will be included here under `fields`.
2. A JSON event/record per line.

See an example in our TCP JSON Source topic.

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

12.3.36. Wavefront Destination

Cribl Edge supports sending events to Wavefront analytics.



Type: Streaming | TLS Support: Yes | PQ Support: Yes

Configure Cribl Edge to Output to Wavefront

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this Wavefront definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
    - **Description**: Optionally, enter a description.
    - **Domain name**: WaveFront domain name, for example, `longboard`. Required.
3. In the **Authentication** settings, select one of the following options from the menu:
    - **Manual**: Displays an **API key** field for you to enter your Wavefront API auth token. See Wavefront's Generating an API Token topic.
    - **Secret**: This option exposes an **Auth token (text secret)** drop-down, in which you can select a stored secret that references the API auth token described above. A **Create** link is available to store a new, reusable secret.
4. Next, you can configure the following **Optional Settings**:
    - **Backpressure behavior**: Select whether to block, drop, or queue events when all receivers are exerting backpressure. (Causes might include a broken or denied connection, or a rate limiter.) Defaults to `Block`.
    - **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

5. Optionally, configure any Persistent Queue, Processing, Retries, and Advanced settings outlined in the sections below.
6. Select **Save**, then **Commit & Deploy**.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.



If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5` GB. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.

Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Retries

Honor Retry-After header: Toggle on to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream/Edge limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. Any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. Toggle off to ignore all `Retry-After` headers.

Default Retry Behavior: By default, Cribl Stream/Edge applies predefined retry strategies for certain failed requests to the Destination using an exponential backoff algorithm. You can override these settings by manually configuring the retry behavior for a particular error code.

Destinations created before the 4.16.0 release do not support the default `4xx` series retry behavior. If your Destination was created prior to this release and you'd like it to support the behavior, reconfigure the Destination.

To retry requests, Cribl Stream/Edge applies the following rules by default:

Status Code	Action
Any in the `1xx`, `3xx` series	Drop the request

Status Code	Action
`401`, `403`, `408`, `429` errors	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached
Any other `4xx` series	Drop the request
Any in the `5xx` series	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached

By default, any `429 (Too Many Requests)` response that is not listed in the retry settings is treated as a non-retryable `4xx` error and dropped. If you want `429` responses retried, ensure `429` is included in the **Settings for failed HTTP requests**.

Settings for failed HTTP requests: You can also manually configure Cribl Stream/Edge to automatically retry requests that receive particular HTTP response status codes.

Upon receiving a response code that's on the list, Cribl Stream/Edge first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream/Edge continues retrying the request without increasing the time between retries any further.

If the sender (which manages the connection to the Destination) is at capacity, it will not accept any incoming events. These incoming events originate internally from a previous stage of the data flow when Destinations send outbound requests to their respective external services, and they include retry requests and new requests. Any events that were already in transit when the sender reached capacity will continue to be processed downstream.

Sender capacity is freed up when an outgoing request succeeds or encounters a non-retryable error. When the sender has available capacity again, it will resume accepting incoming events. This capacity management is influenced by the number of active connections and configured limits, such as concurrency and buffer sizes. If a Pipeline sends events faster than the Destination can process, the buffers may fill up, leading to backpressure and `Sender at capacity` warnings. This backpressure prevents the sender from accepting additional requests until capacity is restored.

For each response code you want to add to the list, select **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.

- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: Toggle on to automatically retry requests that have timed out and display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Toggle on to reject certificates that are **not** authorized by a CA in the **CA certificate path**, nor by another trusted CA (for example, the system's CA).

Round-robin DNS: Toggle on to enable round-robin DNS lookup across multiple IP addresses, IPv4 and IPv6. When a DNS server resolves a Fully Qualified Domain Name (FQDN) to multiple IP addresses, Cribl Edge will sequentially use each address in the order they are returned by the DNS server for subsequent connection attempts.

Compress: Toggle on (default) to compress the payload body before sending.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests per Worker Process. When Cribl Edge hits this limit, it begins throttling traffic to the downstream service. Defaults to `5`. Minimum: `1`. Maximum: `32`.

Body size limit (KB): Maximum size of the request body before compression. Defaults to `4096` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Body size limit** value until downstream receivers reliably accept all events.

Flush period (sec): Maximum time between requests. Low values can cause the payload size to be smaller than the configured **Body size limit**. Defaults to `1` second.

Extra HTTP headers: Click **Add Header** to insert extra headers as **Name/Value** pairs. Values will be sent encrypted.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Cribl Edge will redact all headers, except non-sensitive headers that you declare below in **Safe headers**.

Safe headers: Add headers to declare them as safe to log in plaintext. (Sensitive headers such as `authorization` will always be redacted, even if listed here.) Use a tab or hard return to separate header names.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Proxying Requests

If you need to proxy HTTP/S requests, see System Proxy Configuration.

Notes About Wavefront

For details on integrating with Wavefront, see these Wavefront resources:

- Direct Data Ingestion, and adjacent topics on Wavefront Proxies.
- Wavefront Data Format.

Troubleshooting

The Destination's configuration modal has helpful tabs for troubleshooting:

Live Data: Try capturing live data to see real-time events as they flow through the Destination. On the **Live Data** tab, click **Start Capture** to begin viewing real-time data.

Logs: Review and search the logs that provide detailed information about the delivery process, including any errors or warnings that may have occurred.

Test: Ensures that the Destination is correctly set up and reachable. Verify that sample events are sent correctly by clicking **Run Test**.

You can also view the Monitoring page that provides a comprehensive overview of data volume and rate, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

12.3.37. Webhook Destination

Cribl Edge can send log and metric events to webhooks and other generic HTTP endpoints.



Type: Streaming | TLS Support: Configurable | PQ Support: Yes

Configure Cribl Edge to Output via HTTP

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Destination** and select the Destination you want from the list, choosing either **Select Existing** or **Add New**.
    - To configure via the Routes, select **Data** > **Destinations** or **More** > **Destinations** (Edge). Select the Destination you want. Next, select **Add Destination**.
2. In the **New Destination** modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this HTTP output definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
    - **Description**: Optionally, enter a description.
    - **Load balancing**: Toggle on to specify multiple Webhook URLs and load weights. If toggled off and you notice that Cribl Edge is not sending data to all possible IP addresses, enable **Advanced Settings** > **Round-robin DNS**.
    - **Webhook URL**: Endpoint URL to send events to. For details, see Webhook URLs below.
3. Next, you can configure the following **Optional Settings**:
    - **Exclude current host IPs**: Appears when **Load balancing** is toggled on. It determines whether to exclude all IPs of the current host from the list of any resolved hostnames. Default is toggled off, which keeps the current host available for load balancing.
    - **Method**: The HTTP verb to use when sending events. Defaults to `POST`. Change this to `PUT` or `PATCH` where required by the endpoint.
    - **Format**: The format in which to send out events. The choices are: `NDJSON`, `JSON Array`, `Custom`, and `Advanced`. For details, see Format below.
    - **Backpressure behavior**: Whether to block, drop, or queue events when all receivers are exerting backpressure.

- **Tags**: Optionally, add tags that you can use to filter and group Destinations on the **Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

4. Optionally, you can adjust the TLS, Authentication, Persistent Queue Settings, Processing, and Advanced settings outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

Webhook URLs

Use the **Webhook URLs** table to specify a known set of receivers on which to load-balance data. To specify more receivers on new rows, click **Add URL**. Each row provides the following fields:

Webhook LB URL: Specify the Webhook URL to send events to - for example, `http://webhook:8000/`

Load weight: Set the relative traffic-handling capability for each connection by assigning a weight (`> 0`). This column accepts arbitrary values, but for best results, assign weights in the same order of magnitude to all connections. Cribl Edge will attempt to distribute traffic to the connections according to their relative weights.

The final column provides an `X` button to delete any row from the table.

For details on configuring all these options, see About Load Balancing.



When you first enable load balancing, or if you edit the load weight once your data is load-balanced, give the logic time to settle. The changes might take a few seconds to register.

Format

When sending events to a Webhook Destination, you can choose the format in which they are delivered.

- `NDJSON`: Newline-delimited JSON (the default).
- `JSON Array`: Arrays in JSON-parseable format.
- `Custom`: Events and event batches (payloads) whose format you define using JavaScript **expressions**. See Custom Format below.
- `Advanced`: Events and event batches (payloads) whose format you define using JavaScript **code**. Enables the Webhook Destination to integrate with APIs not otherwise supported in Cribl Edge. See Advanced Format below.

TLS Settings (Client Side)

Enabled Default is toggled off. When toggled on:

Server name (SNI): Server name for the SNI (Server Name Indication) TLS extension. This must be a host name, not an IP address.

Minimum TLS version: Optionally, select the minimum TLS version to use when connecting.

Maximum TLS version: Optionally, select the maximum TLS version to use when connecting.

Certificate: Select an existing certificate name from the drop-down or add a new certificate by selecting **Create**. See Add a TLS Certificate to a Fleet for details.

CA certificate path: Path on client containing CA certificates (in PEM format) to use to verify the server's cert. Path can reference `$ENV_VARS`.

Private key path (mutual auth): Path on client containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Certificate path (mutual auth): Path on client containing certificates in (PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Passphrase: Passphrase to use to decrypt private key.

Authentication

Select one of the following options for authentication:

- **None**: Don't use authentication.
- **Auth token**: Use HTTP token authentication. In the resulting **Token** field, enter the bearer token that must be included in the HTTP authorization header.
- **Auth token (text secret)**: This option exposes a **Token (text secret)** drop-down, in which you can select a stored text secret that references the bearer token described above. A **Create** link is available to store a new, reusable secret.
- **Basic**: Displays **Username** and **Password** fields for you to enter HTTP Basic authentication credentials.
- **Basic (credentials secret)**: This option exposes a **Credentials secret** drop-down, in which you can select a stored text secret that references the Basic authentication credentials described above. A **Create** link is available to store a new, reusable secret.
- **OAuth**: Configure authentication via the OAuth protocol. Exposes multiple extra options - see OAuth Authentication below.

OAuth Authentication

Selecting **OAuth** exposes the following additional fields:

- **Login URL**: Endpoint for the OAuth API call, which is expected to be a POST call.

- **OAuth secret**: Secret parameter value to pass in the API call's request body.
- **OAuth secret parameter name**: Secret parameter name to pass in API call's request body.
- **Token attribute name**: Name of the auth token attribute in the OAuth response. Can be top-level (for example, `token`); or nested, using a period (for example, `data.token`).
- **Authorize expression**: JavaScript expression used to compute the Authorization header to pass in requests. Uses `${token}` to reference the token obtained from the login POST request, for example: `` `Bearer ${token}` ``.
- **Refresh interval (secs.)**: How often to refresh the OAuth token. Default `3600` seconds (1 hour); minimum `1` second; maximum `300000` seconds (about 83 hours).
- **OAuth parameters**: Optionally, select **Add parameter** for each additional parameter you want to send in the OAuth login request. Cribl Edge will combine the secret with these parameters, and will send the URL-encoded result in a POST request to the endpoint specified in the **Login URL**. We'll automatically add the `Content-Type` header `application/x-www-form-urlencoded` when sending this request.
  In each row of the resulting table, enter the parameter's **Name**. The corresponding **Value** is a JavaScript expression to compute the parameter's value. This can also evaluate to a constant. Values not formatted as expressions - for example, `id` instead of `` `${id}` `` - will be evaluated as strings.
- **Authentication headers**: Optionally, select **Add Header** for each custom auth header you want to define and send in the OAuth login request. Stream will automatically add the `Content-Type` header `application/x-www-form-urlencoded` when sending this request.
  In each row of the resulting table, enter the custom OAuth header's **Name**. The corresponding **Value** is a JavaScript expression to compute the header's value. This can also evaluate to a constant. Values not formatted as expressions - for example, `id` instead of `` `${id}` `` - will be evaluated as strings.

Persistent Queue Settings

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

> This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

> If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5 GB`. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are currently queued for delivery to this Destination. If you click this button, a confirmation modal appears. Clearing the queue frees up disk space by permanently deleting the queued data, without delivering it to downstream receivers. This button only appears after you define the **Output ID**.



Use the **Clear Persistent Queue** button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

Processing Settings Post-Processing

Pipeline: Pipeline or Pack to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Edge Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` - Cribl Edge Node that processed the event.
- `cribl_input` - Cribl Edge Source that processed the event.
- `cribl_output` - Cribl Edge Destination that processed the event.
- `cribl_route` - Cribl Edge Route (or QuickConnect) that processed the event.
- `cribl_wp` - Cribl Edge Worker Process that processed the event.

Retries

Honor Retry-After header: Toggle on to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream/Edge limits the delay to 180 seconds even if the `Retry-After`

header specifies a longer delay. Any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. Toggle off to ignore all `Retry-After` headers.

Default Retry Behavior: By default, Cribl Stream/Edge applies predefined retry strategies for certain failed requests to the Destination using an exponential backoff algorithm. You can override these settings by manually configuring the retry behavior for a particular error code.

To retry requests, Cribl Stream/Edge applies the following rules by default:

Status Code	Action
Any in the `1xx, 3xx` series	Drop the request
`408, 429` errors	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached
Any other `4xx` series	Drop the request
Any `5xx` series	Retry in memory or persistent queue if enabled, and initiate backpressure after limit is reached

By default, any `429 (Too Many Requests)` response that is not listed in the retry settings is treated as a non-retryable `4xx` error and dropped. If you want `429` responses retried, ensure `429` is included in the **Settings for failed HTTP requests**.

Settings for failed HTTP requests: You can also manually configure Cribl Stream/Edge to automatically retry requests that receive particular HTTP response status codes.

Upon receiving a response code that's on the list, Cribl Stream/Edge first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream/Edge continues retrying the request without increasing the time between retries any further.

If the sender (which manages the connection to the Destination) is at capacity, it will not accept any incoming events. These incoming events originate internally from a previous stage of the data flow when Destinations send outbound requests to their respective external services, and they include retry requests and new requests. Any events that were already in transit when the sender reached capacity will continue to be processed downstream.

Sender capacity is freed up when an outgoing request succeeds or encounters a non-retryable error. When the sender has available capacity again, it will resume accepting incoming events. This capacity management is influenced by the number of active connections and configured limits, such as concurrency and buffer sizes. If a Pipeline sends events faster than the Destination can process, the buffers may fill up, leading to backpressure

and `Sender at capacity` warnings. This backpressure prevents the sender from accepting additional requests until capacity is restored.

For each response code you want to add to the list, select **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: Toggle on to automatically retry requests that have timed out and display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream/Edge will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream/Edge should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Reject certificates that are **not** authorized by a CA in the **CA certificate path**, nor by another trusted CA (for example, the system's CA). Defaults to toggled on.

Round-robin DNS: Toggle on to enable round-robin DNS lookup across multiple IP addresses, IPv4 and IPv6. When a DNS server resolves a Fully Qualified Domain Name (FQDN) to multiple IP addresses, Cribl Edge will sequentially use each address in the order they are returned by the DNS server for subsequent connection attempts.

Compress: Toggle on (default) to compress the payload body before sending.

Keep alive: By default, Cribl Edge sends `Keep-Alive` headers to the remote server and preserves the connection from the client side up to a maximum of 120 seconds. Toggle this off if you want Cribl Edge to close the connection immediately after sending a request.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests per Worker Process. When Cribl Edge hits this limit, it begins throttling traffic to the downstream service. Defaults to `5`. Minimum: `1`. Maximum: `32`.

Body size limit (KB): Maximum size of the request body before compression. Defaults to `4096` KB. You can set this limit to as high as 500 MB (`512000` KB). Be aware that high values can cause high memory usage per Worker Node, especially if a dynamically constructed URL (see Internal Fields) causes this Destination to send events to more than one URL. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Body size limit** value until downstream receivers reliably accept all events.

Buffer memory limit (KB): Total amount of memory used to buffer outgoing requests waiting to be sent. If left blank, defaults to 5 times the max body size (if set). If 0, no limit is enforced. This provides granular control over the memory allocated for buffering outgoing batched requests. Increasing the limit allows batches to grow larger before being flushed, improving efficiency for data with high cardinality (a large number of unique batches). Finding the optimal balance between efficient data transfer and memory usage involves adjusting both the **Buffer memory limit** and **Body size limit** settings.

Events-per-request limit: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Extra HTTP headers: Name-value pairs to pass to all events as additional HTTP headers. Values will be sent encrypted. You can also add headers dynamically on a per-event basis in the `__headers` field. See Internal Fields below.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Cribl Edge will redact all headers, except non-sensitive headers that you declare below in **Safe headers**.

Safe headers: Add headers to declare them as safe to log in plaintext. (Sensitive headers such as `authorization` will always be redacted, even if listed here.) Use a tab or hard return to separate header names.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Custom Format

Choosing `Custom` from the **General Settings** > **Format** drop-down exposes the additional fields described in this section. **Source expression** and most of the rest of the controls define the format of individual events - that

is, what data you want to make available from the event. **Batch expression** defines the wrapper object for batched events - that is, how to package events within the request payload. Both expressions **must** be enclosed in backticks.

- **Source expression**: JavaScript expression to evaluate on every event; Cribl Edge will send the result of that evaluation instead of the original event. Sample expression: `` `${fieldA}, ${fieldB}` `` (with literal backticks). Defaults to `__httpOut` - that is, the value of the `__httpOut` field. Use the button at right to open a validation modal.
- **Drop when null**: If toggled on, Cribl Edge will drop events when the **Source expression** evaluates to `null`.
- **Event delimiter**: Delimiter string to insert between events. Defaults to the newline character (`\n`). Cannot be a space (this will be converted to `\n`).
- **Content type**: Content type to use for requests. Defaults to `application/x-ndjson`. Any content types set in **Advanced Settings** > **Extra HTTP headers** will override this entry.
- **Batch expression**: Expression specifying how to format the payload for each batch. Defines a wrapper object in which to include the formatted events, such as a JSON document. This enables requests to APIs that require such objects. To reference the events to send, use the `${events}` variable. An example expression to send the batch inside a JSON object would be: `{"items" : [${events}] }`.

Custom Format Example

Suppose that the API you're sending events to requires request payloads in the following form:

```
{
   "Events":[
      {
         "one":"1",
         "two":"2"
      },
      {
         "one":"1.1",
         "two":"2.2"
      }
   ]
}
```

Copy

And here's how the events coming into your Webhook Destination look - note that the field names need to change, and `field3` and `field4` need to be dropped:

```
{ "field1": 1, "field2": 2, "field3": 3, "field4": 4 }
{ "field1": 1.1, "field2": 2.2, "field3": 3.3, "field4": 4.4 }
```

Copy

To send these events in the required form, you'd do the following:

- Set the **Source expression** to `` `{ "one": "${field1}", "two": "${field2}" }` `` - note that this renames and sends only the desired fields (`one` and `two`).
- Enter the comma (`,`) as your **Event delimiter**.
- Set the **Batch expression** to `` `{ "Events": [${events}] }` ``.

Advanced Format

Choosing `Advanced` from the **General Settings** > **Format** drop-down exposes the JavaScript code blocks described in this section. You can define a format using either one of the code blocks, or both.

- **Format inbound event**: JavaScript code block that receives incoming events (as JavaScript objects) and converts them into strings. Multiple strings are concatenated to form a batch, the batches constituting the payloads of the HTTP requests that the Destination sends to the downstream API.
- **Format outbound payload**: JavaScript code block that receives batches before they are sent out as HTTP request payloads. This gives you the opportunity to modify the payload format from the way it arrives - as a string of concatenated events - to whatever format the downstream API requires.

Code Block Restrictions

Generally speaking, anything forbidden in JavaScript strict mode is forbidden in the code blocks. Specifically, the following are **not allowed**:

- `console`, `eval`, `uneval`, `Function (constructor)`, `Promises`, `setTimeout`, `setInterval`, `global`, `globalThis`, `window`, and `set`.

Code blocks **can** include `for` loops, `while` loops, and JavaScript methods such as `map`, `reduce`, `forEach`, `some`, and `every`. For further details, see Supported JavaScript Options.

Only skilled JavaScript developers should use the Advanced format. This is to avoid unintended results - such as creating infinite loops, or otherwise failing to return - that could needlessly add to your throughput burden.

Advanced Format Syntax

Here are the syntax conventions for manipulating events and payloads within the code blocks.

Format inbound event Syntax

- `__e`: References the event object.
- `__e['fieldName']`: References an individual field in the event object.
- `__e.asJSON(): string`: Method to safely convert the object to JSON by removing internal content that is not relevant for a downstream system. **Always** use this method instead of directly converting the object to JSON string using `JSON.stringify(__e)`. This avoids circular references that can cause `JSON.stringify(__e)` to fail.

Format outbound event Syntax

In addition to the syntax conventions defined above, the **Format outbound event** code block has its own special conventions:

- `__e['__payload']`: Accesses the `payload` attribute of the JavaScript object that the code block receives. The `__payload` attribute contains the actual batch (payload) to be formatted.
- `__e['__payloadOut']`: After you modify the batch (payload) as desired, assign it to this variable. If the `__payloadOut` attribute is not present, the code block sends the payload as-is. See the JSON Array Example below.

Advanced Format Process

The **Format inbound event** and **Format outbound payload** code blocks are populated by placeholder JavaScript code. This code expresses the basic flow you must understand to define an Advanced format. Let's examine the code - then you'll be able to see how defining any given Advanced format is just a matter of manipulating events and payloads within the basic flow.



The function definitions `formatEvent (__e : CriblEvent)` and `formatPayload (__e : CriblEvent)` are shown for reference. Do not include them in your code blocks. You only need to enter the internal JavaScript code.

The Format inbound event Code Block

The event to format appears in the first line, referenced as `__e: CriblEvent`. First, you serialize the event as a string; the simplest way to do that is with the method `e.asJSON()`. Then, you assign it back to the object in a variable named `__eventOut`.

```
formatEvent (__e : CriblEvent) {
  // TODO: Serialize the event to a string here.
  let formattedEvent = __e.asJSON();

  // Assign formatted event to __eventOut.
  __e['__eventOut'] = formattedEvent;
}
```

Copy

At this point, the `formatEvent()` function ensures that each incoming event will be serialized as a string. (In the more elaborate examples below, the format of that string will be manipulated.) Each new event, in string form, is appended to the last, building a longer string that will become a **batch**. How many events can be in a batch is governed by **Advanced Settings** > **Events-per-request limit**.

Once a batch is complete, the next code block can operate on it.

The Format outbound payload Code Block

Now the `formatPayload ()` function receives an event whose `payload` attribute is the batch produced by the previous code block. You assign that attribute to the `formattedPayload` variable. If you want to manipulate the batch, do that here. Once you have transformed the batch, you assign it to the `__payloadOut` variable, and the Webhook Destination sends the batch out as the payload of an HTTP request.

Transforming batches is not required. To send batches out exactly how they leave the **Format inbound event** code block, just leave the **Format outbound payload** code block blank.

```
formatPayload: (__e : CriblEvent) {
  // Access payload to format here
  let formattedPayload = __e['payload']

  // TODO: Format payload here

  // Assign formatted payload to payloadOut
  __e['payloadOut'] = formattedPayload
}
```

Copy

JSON Array Example

Suppose you want to communicate with an API that accepts data in JSON array format.

Use the **Format inbound event** code block to format the events:

```
__e['__eventOut'] = `${__e.asJSON()},` // Serialize to JSON and add trailing comma
```

Copy

Use the **Format outbound event** code block to format the payload:

```
// Remove trailing comma and wrap payload in array
__e['__payloadOut'] = `[${payload.substring(0,payload.length-1)}]`
```

Copy

Elasticsearch Bulk API Example

The Elasticsearch Bulk API requires each event to be formatted with two lines of output. While this cannot be done with the Webhook Destination's other format options, it is doable with Advanced Format. (The Elasticsearch API Source, Elasticsearch Destination and Elastic Cloud Destination **do** work with the Elasticsearch Bulk API.)

Here's how Bulk API events must be formatted, structurally:

```
{ <action>:  { <metadata> } }
{ <serializedEvent> }
```

Copy

Here's an example of an event formatted for Bulk API:

```
{ "index" : { "_index" : "test", "_id" : "1" } }
{ "field1" : "value1", "field2" "value2", "field3" : "value3" }
```

Copy

Use the **Format inbound event** code block to format the events:

```
let time=__e['_time'] || Date.now();
__e['_time'] = undefined;
__e['@timestamp'] = C.Time.strftime(time, '%Y-%m-%dT%H:%M:%S.%LZ');
if(typeof __e['host'] === 'string') {
  const host = __e['__host'] || {};
  host.name = __e['host'];
  __e['host'] = host;
}
__e['__eventOut'] = `{"create":{"_index":"${__e['_index'] || 'cribl'}", "pipeline":"$
```

Copy

Leave the **Format outbound event** code block blank.

Splunk HEC Example

Use the **Format inbound event** code block to format the events:

```
  const eventIn = __e;
  const time = eventIn._time;
  const event = eventIn._raw;
  const {host, source, sourcetype, index} = eventIn;
  eventIn._time = undefined;
  eventIn._raw = undefined;
  eventIn.host = undefined;
  eventIn.source = undefined;
  eventIn.sourcetype = undefined;
  eventIn.index = undefined;
  const fields = {};
  const obj2send = {time, index, host, source, sourcetype, event, fields};
  for (const key of ['index', 'host', 'source', 'sourcetype']) {
    // only convert if the key/val is present and is not a string
    const oldValue = obj2send[key];
    if (oldValue != null && typeof oldValue !== 'string') {
      obj2send[key] = JSON.stringify(oldValue);
    }
  }
  for (let key of Object.keys(eventIn)) {
    if(key.startsWith('__'))
      continue;
    let val = eventIn[key];
    if(val === undefined)
      continue;
    if(typeof val === 'object')
      val = JSON.stringify(val);
    obj2send.fields[key] = val;
  }
__e['__eventOut'] = `${JSON.stringify(obj2send)}\n`;
```

Copy

Leave the **Format outbound event** code block blank.

Internal Fields

Cribl Edge uses a set of internal fields to assist in forwarding data to a Destination.

Fields for this Destination:

- `__criblMetrics`

- `__url`
- `__headers`

If an event contains the internal field `__criblMetrics`, Cribl Edge will send it to the HTTP endpoint as a metric event. Otherwise, Cribl Edge will send it as a log event.

When **Load balancing** is disabled, if an event that contains the internal field `__url`, that field's value will override the **General Settings** > **Webhook URL** value. This way, you can determine the endpoint URL dynamically.

For example, you could create a Pipeline containing an Eval Function that adds the `__url` field, and connect that Pipeline to your Webhook Destination. Configure the Eval Function to set `__url`'s value to a URL that varies depending on a global variable, or on some property of the event, or on some other dynamically-generated value that meets your needs.

If an event contains the internal field `__headers`, that field's value will be a JSON object containing Name-value pairs, each of which defines a header. By creating Pipelines that set the value of `__headers` according to conditions that you specify, you can add headers dynamically.

For example, you could create a Pipeline containing Eval Functions that add the `__headers` field, and connect that Pipeline to your Webhook Destination. Configure the Eval Functions to set `__headers` values to Name-value pairs that vary depending on some properties of the event, or on dynamically-generated values that meet your needs.

Here's an overview of how to add headers:

- To add "dynamic" (a.k.a. "custom") headers to some events but not others, use the `__headers` field.
- To define headers to add to **all** events, use **Advanced Settings** > **Extra HTTP Headers**.
- An event can include headers added by both methods. So if you use `__headers` to add `{ "api-key": "foo" }` and **Extra HTTP Headers** to add `{ "goat": "Kid A" }`, you'll get both headers.
- Headers added via the `__headers` field take precedence. So if you use `__headers` to add `{ "api-key": "foo" }` and **Extra HTTP Headers** to add `{ "api-key": "bar" }`, you'll get only one header, and that will be `{ "api-key": "foo" }`.

Use Cases

See these examples of configuring a Webhook Destination to integrate with specific services:

- Azure Analytics/Sentinel Integration
- Webhook/BigPanda Integration
- Webhook/Slack Integration

Notes on HTTP-Based Outputs

12.4. Manage Backpressure

Backpressure refers to mechanisms that manage the flow of data when production outpaces consumption. Sometimes an upstream Source can cause a backpressure event when Cribl Edge receives a sudden influx of data that it can't immediately process. And sometimes a downstream Destination can cause a backpressure event when the receiver doesn't have the capacity to receive incoming data from Cribl Edge. Backpressure events carry the risk of data loss, system crashes, or other undesirable effects.

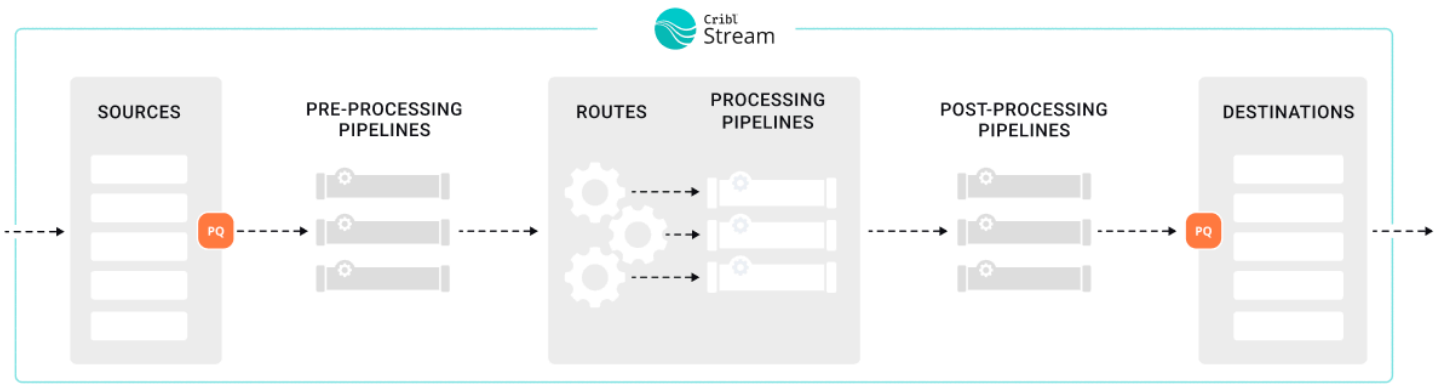
Fortunately, Cribl Edge has built-in methods for responding to backpressure events. It has feedback mechanisms and triggers for detecting backpressure events. Cribl Edge also includes various settings for responding to backpressure events when they occur, such as enabling persistent queueing to store data during backpressure events.

Read the pages in this section to learn more about how Cribl Edge handles backpressure and backpressure configuration options:

- About Persistent Queues
- Optimize Source Persistent Queues
- Optimize Destination Persistent Queues
- About Destination Backpressure Triggers
- Backpressure Impacts to Sources

12.4.1. About Persistent Queues

Cribl Edge's persistent queues (PQ) supplement in-memory queues to minimize data loss during a backpressure event. When a backpressure event occurs, Cribl Edge can engage a backpressure queue to temporarily store data by writing it to disk during the outage. When the outage resolves, Cribl Edge then forwards data from the queue.



You can enable persistent queues at both ends of the data flow

Cribl Edge's persistent queue (PQ) feature helps minimize data loss in these situations:

- A downstream Destination receiver is unreachable.
- A downstream Destination receiver is slow to respond. (This feature is only available in Cribl Edge 4.9.0 and newer.)
- When a configured Source sends a higher volume of data than Cribl Edge can process.

Persistent queues are best in environments that need more durability and reliability. Persistent queues might be a good fit if:

- Your environment regularly experiences outages or backpressure events that last longer than short-term memory queues can sustain, which is typically only a few seconds.
- Your business or organization has a low level of tolerance for data loss.
- You need data from upstream senders that do not provide their own buffering for backpressure queues, such as UDP senders.

You can use persistent queues on:

- Push Sources
- Streaming Destinations

See Supported Sources and Supported Destinations for more information.

The effectiveness of persistent queues to minimize data loss depends on your deployment type, configuration settings, and how much disk space you have allocated for persistent queues. For more information about how to optimize and tune your persistent queue settings, see:

- Optimize Source Persistent Queues
- Optimize Destination Persistent Queues

The persistent queue feature is only available for certain Cribl plans or license types. See the Pricing page for details.

How Persistent Queues Work

The persistent queue feature works differently on Sources than it does on Destinations. For information about how it works, see:

- How Source Persistent Queues Work
- How Destination Persistent Queues Work

When to Use Persistent Queues

When deciding whether to use persistent queueing or not, these factors should guide your decisions:

Cloud or Self-Managed (On-Prem) Deployments

If your environment is a Cribl-managed Cribl.Cloud deployment, Cribl manages the performance and tuning of your cloud deployment. For that reason, there is not a clear downside or tradeoff for enabling persistent queue. Instead, ensure you understand how persistent queues work for both Sources and Destinations and then make a decision about whether to enable it or not. For more information, see:

- Source persistent queue: For Cribl.Cloud Only: Whether to Enable Source Persistent Queue
- Destination persistent queue: For Cribl.Cloud Only: Whether to Use a Hybrid Group

For self-managed (on-prem) deployments, evaluate the additional factors described on this page when making a decision about implementing persistent queues.

Support for Backpressure Queues on Senders and Receivers

For both cloud and on-prem deployments, take some time to learn about the upstream and downstream senders and receivers for your data. Do these senders and receivers support backpressure queues? For

example, upstream senders that are UDP-based do not support backpressure queues, but TCP-based Destinations tend to have backpressure queue support.

If the upstream sender or downstream receiver supports backpressure queueing, determine whether to use the native backpressure queue features in those senders and receivers instead of the Cribl Edge persistent queues. In general, you should only use backpressure queues on either Cribl Edge *or* the sender/receiver. Using them both together on both ends of transmission is redundant and can result in data latency issues. However, latency is a natural consequence of queueing, so you need to balance your needs for data resiliency against your needs for timeliness.



What is the difference between a persistent queue and a backpressure queue?

Persistent queues are specifically on-disk storage. Backpressure queues can be either in-memory or on-disk buffers. Most systems have some type of backpressure queueing, which is not quite the same as the persistent queue feature provided by Cribl. When you integrate a sender or receiver to Cribl Edge, ensure you research and understand the options provided by those senders and receivers. Carefully consider how these two systems might interact together.

Likelihood of Outages and Backpressure

For self-managed (on-prem) deployments, consider the overall likelihood of outages or other backpressure events that could result in data loss:

- Does your system regularly experience outages or backpressure?
- What is the duration of these outages or events?
- Do the outages or events often last longer than short-term memory queues can sustain?

If so, then persistent queueing might be the right choice for your system.

Risk Tolerance for Data Loss

For self-managed (on-prem) deployments, determine whether your organization has a high or low risk tolerance for data loss. If they have a high need for data integrity, persistent queueing might be appropriate.

Infrastructure Costs Needed for Persistent Queue

For self-managed (on-prem) deployments, you may need to balance your business requirements against the cost of maintaining the resources and infrastructure necessary to engage persistent queues to prevent data loss. Persistent queues incur costs related to processing, computing, and storage because you have to write the data to disk.

Persistent queues can be resource-intensive for Worker Processes. When enabled, each Worker Process maintains a queue for both Source and Destination persistent queues. Depending on your configuration and the frequency of outages, Worker Processes may allocate significant processing power and compute resources to managing these queues.

Persistent queues also introduce overhead when they engage. Writing to or reading from the disk is slower and demands CPU resources. For instance, using Source persistent queues in Smart Mode can increase CPU overhead if the queue frequently engages and disengages during operation.

Fortunately, performance tuning your persistent queue system could help reduce costs. For more information, see:

- Optimize Source Persistent Queues
- Optimize Destination Persistent Queues

Persistent Queue Limitations and Constraints

Persistent queues:

- Operate at the Worker Process level, with each Worker Process sharing the same configuration settings for a given Source or Destination. These settings are inherited from the respective Source or Destination. The size and engagement model of the queue may vary depending on the specific Source or Destination.
- Have a finite size. If the system cannot deliver data, it might eventually exhaust available disk space.
- Provide limited data protection in cases of application failure. For example, a crash may cause the loss of in-memory data.
- Offer limited data protection in cases of hardware failure, such as disk failure, data corruption, or machine/host loss.



If you turn off persistent queue for a Source or Destination (or shut down the whole Source or Destination) before its queues drain, you will orphan any events still queued on disk.

The following constraints apply to certain Destinations:

- With load-balanced Destinations, persistent queue engages only when *all of the Destination's receivers* are slow or blocking data flow. A single live receiver will prevent persistent queue from engaging on the corresponding Destination. Persistent queue will not engage if at least one outbound TCP socket connection is active, even if the receiver is sending a backpressure signal.
- For Kafka-based Destinations (such as Kafka, Confluent Cloud, Amazon MSK, and Azure Event Hubs), persistent queues engage when Cribl Edge can't reach the brokers.
- The system controls HTTP/S-based Destinations to prevent retries of failed events. However, some events in transit to the Destination may still emerge as duplicates.

Backpressure Behavior with Multiple Destinations

When Cribl Edge sends the same event to multiple Destinations, the resulting behavior depends on the backpressure setting for each Destination:

- **Block**: If any Destination is configured to Block and enters backpressure, Cribl Edge blocks delivery for that event across all Destinations that receive the same input data.
- **Persistent Queue**: If a Destination is configured to use Persistent Queue, Cribl Edge can queue that copy of the event locally while healthy Destinations continue to receive data, as long as the queue has available capacity.
- **Queue-full**: If the queue becomes full, Cribl Edge follows the configured **Queue-full behavior** for that Destination, which can block or drop new data.

This behavior applies whenever the same event is sent to multiple Destinations, including:

- Fan-out patterns via QuickConnect to multiple Destinations
- Multiple Routes with **Final** toggled off
- Output Router rules with **Final** toggled off

For load-balanced Destinations, persistent queue engages only when all receivers are slow or blocking. A single healthy receiver prevents persistent queue from engaging on that Destination.

Supported Sources

Cribl Edge supports persistent queueing for Push Sources. These sources ingest streaming data, which makes it possible to enable persistent queue.

Persistent queueing is not available for Pull and Collector Sources, which ingest data intermittently and/or from static files.

To determine whether Cribl Edge supports persistent queue for a specific Source, consult the documentation for that Source.

Supported Destinations

Cribl Edge supports persistent queue for:

- HTTP-based Destinations
- TCP Load-Balanced Destinations
- Destinations that write to a single receiver (without load balancing enabled)

To determine whether Cribl Edge supports persistent queue for a specific Destination, consult the documentation for that Destination.

12.4.2. Optimize Source Persistent Queues

Cribl Edge's Source persistent queues can supplement in-memory queues to prevent data loss during a backpressure event, such as when a configured Source sends a higher volume of data than Cribl Edge can process. See Supported Sources for more information about which Sources support persistent queueing.

Read this guide to learn how to optimize your Source persistent queue. This guide is helpful when you are in the initial planning and configuration stage and also when you have an active deployment that you want to performance tune. This guide explains:

- How Source persistent queues respond to a backpressure event.
- The overall goals and key metrics involved in optimizing Source persistent queues.
- An in-depth discussion of the Source persistent queue configuration options and their potential impact on your system.
- The available notifications, monitoring dashboards, and troubleshooting tools for Source persistent queues.

For a general overview of persistent queues, see About Persistent Queues.

> On Cribl-managed Cribl.Cloud Workers with an Enterprise plan, you can enable persistent queue for supported Sources. However, be aware that mode options and storage limits have specific restrictions in Cribl.Cloud. See For Cribl.Cloud Only: `Always On` is the Only Available Mode for more information.

How Source Persistent Queues Work

Backpressure occurs when there is a temporary imbalance between the rates of inbound and outbound data that is either coming in or going out of Cribl Edge.

Different events trigger persistent queues on Sources:

- Error conditions.
- Degraded performance–such as slowdowns or short backups–in downstream Cribl resources.
- External receivers might trigger backpressure.

When Source persistent queue engages, it writes and stores data on a local file system in your environment until the backpressure event resolves.

How Sources Handle Backpressure

Source persistent queue can serve as a stopgap for upstream senders that don't provide their own buffering, or whose small buffers cannot queue for long. This applies to several syslog senders and HTTP-based Sources.

Sources that have persistent queue support have the option to supplement in-memory buffering with persistent queue. When you enable persistent queue, you can choose between these Source persistent queue trigger conditions:

Condition	Description
Always On	Uses persistent queue as a disk buffer for all events.
Smart	Only engages persistent queue upon backpressure from downstream receivers or Cribl internal processes.

In `Always On` mode, the persistent queue constantly writes all data to a backup disk buffer regardless of whether there is a backpressure event or normal conditions. After the backpressure event ends, the persistent queue flushes every 10 seconds when it drains in `Always On` mode.

In `Smart` mode, persistent queue engages under any of these conditions:

- A connected Destination is fully down and Destination-side persistent queue is not enabled.
- A connected Destination is slow or blocking.
- A downstream Cribl Pipeline or Route is slow.

When persistent queue engages on a Source in `Smart` mode:

- Events queue *before* any pre-processing Pipeline takes effect.
- Metrics (data in) will show `0` bytes/events while the system buffers events to the disk queue.
- Cribl Edge Pipelines and Routes process the data with a delay as the persistent queue drains.

`Smart` mode only engages when necessary, such as when a downstream Destination becomes blocked *and* the **Buffer size limit** reaches its limit. When persistent queue is set to `Smart` mode, Cribl attempts to flush the queue when every new event arrives. The only time events stay in the buffer is when a downstream Destination becomes blocked.



Enabling persistent queue on a Source is redundant if the upstream sender already safeguards events in its own disk buffer. Enabling persistent queue on both the sender and Cribl Edge could cause data latency. Check the settings for the sender to ensure you know how whether that specific sender already safeguards against data loss.

If you turn off Source persistent queue (or shut down the whole Source) before its queues drain, you will orphan any events still queued on disk.

Special Consideration for UDP Sources

For UDP Sources, downstream blocking is especially risky because UDP senders do not support backpressure signaling. When Cribl Edge cannot process incoming UDP data fast enough due to downstream backpressure or processing bottlenecks, it will drop new events rather than signal the sender to slow down.

To reduce data loss during short-term downstream issues:

1. **Enable Source persistent queue (sPQ)** on UDP-based Sources where supported. This allows Cribl Edge to buffer incoming events to disk when downstream components are slow or blocked.
2. **Increase UDP buffer sizes** in Advanced Settings to handle brief traffic spikes:
    - **Buffer size limit (events)**: Increase this value to buffer more events in memory when downstream is blocking. Default is 1000 events.
    - **UDP socket buffer size (bytes)**: Increase this kernel-level buffer to prevent the OS from dropping events before Cribl Edge can process them.
3. **Monitor for dropped messages** on the Source's **Status** tab. Dropped messages may indicate the need for larger buffers or that downstream pressure is exceeding what buffering can absorb.

For more information, see UDP Tuning.

Configure Source Persistent Queue

You configure Source persistent queue behavior when you configure each individual Source. Consult the documentation for that Source for more information about initial configuration.

This section provides a configuration quick start for experienced users. However, new Cribl Edge users should consider reading the full guidance in this topic to get a deeper understanding of the decisions and ideal system architecture design for persistent queue.

To configure persistent queue on a Source:

1. In the **Persistent Queue Settings** tab, toggle **Enable persistent queue** off. See Whether to Enable Persistent Queues for Backpressure Events for guidance about making the decision to enable persistent queue.
2. For On-Prem only: Select an appropriate **Mode** from the available options. See When and How to Store Data to Disk During a Backpressure Event for information about which mode to select.

On Cribl-managed Cribl.Cloud Workers with an Enterprise plan, you can enable persistent queues for supported Sources. By default, Source persistent queue is not enabled, so you must explicitly decide to turn it on.

When you enable Source persistent queue in Cribl.Cloud, the only available mode is `Always On`, which writes data to disk at all times. `Smart` mode is not available. See For Cribl.Cloud Only: `Always On` is the Only Available Mode for more information.

3. For On-Prem only: Set the **Buffer size limit** to determine the maximum number of events to hold in memory before reporting backpressure to the sender. See When to Report Backpressure to the Sender for more information.
4. For On-Prem only: Set the **Commit frequency** to determine how many events to send downstream before committing that Cribl Stream has read them. See How Many Events to Commit During a Backpressure Event for more information.
5. For On-Prem only: Set the **File size limit** to specify the maximum amount of data stored in each queue file before the file closes and starts a new file. Set the **Queue size limit** to determine the maximum amount of disk space that Cribl allows the queue to consume on each Worker Process. See How Much Data to Store in Memory for more information about these settings.
6. Set the **Queue file path** to determine the location for the persistent queue files. See Where to Store Data and How Much Disk Space to Allocate for more information.
7. Optional: Set the **Compression** type to apply compression to the data before the queue file closes. See Whether to Compress Data for more information.

Overview of Source Persistent Queue Optimization

A well-optimized data throughput system can efficiently and reliably processes and transfer data from Sources to Destinations. Optimized systems maximize performance while minimizing latency, errors, and resource consumption. An optimized system has these characteristics:

- **High efficiency:** Data flows seamlessly at the maximum achievable rate without bottlenecks or excessive resource usage. The system gracefully handles backpressure events, ensuring that data intelligently throttles or queues without overwhelming downstream components.
- **Reliability:** The system consistently handles variations in data volume or speed without losing or corrupting data, even during spikes or failures.
- **Scalability:** The system adapts smoothly to increasing data loads or additional data senders or receivers without degradation in performance.
- **Low latency:** The system processes and delivers data in near real-time, meeting the required time constraints for your business use cases.
- **Cost-efficient resource allocation:** The system uses CPU, memory, disk, and network resources effectively, avoiding over-provisioning or waste.

- **Security and compliance:** The system transmits data securely, meeting all relevant privacy, encryption, and regulatory requirements.

Ultimately, you and your business stakeholders should have confidence in the system's performance, reliability, and ability to adapt to future needs. Persistent queue is an important tool to achieve this confidence.

Persistent Queue Optimization Metrics

The most effective way to performance tune is to understand which key metrics give insights into your persistent queues system's overall health. Effective system administrators study their deployments to establish a performance baseline that they can use for comparison if a performance issue arises. Most system administrators understand their system well and can easily identify peak data periods.

Pay attention to these metrics:

- **CPU utilization:** When your system gets sudden increases in data loads, you might notice that your Worker Processes operate at full capacity and use all their CPU. Generally, that's a sign of inefficiency or the need for more resources. If needed, you can teleport into a single Worker and view all the processes and their individual CPU utilization. To see your CPU utilization, you can use the system monitoring tools to analyze CPU utilization. If needed, you can also teleport into a single Worker and view all the processes and their individual CPU utilization.
- **Data latency:** Some persistent queues can add latency because of the amount of overhead time it takes to write data to disk. As a starting point, assume that each event incurs a few milliseconds of added latency. Be aware that data latency can vary by file system, such as whether you are using a local file system or a Network File System (NFS). NFS file systems tend to introduce higher and more variable latency. To measure the latency impact of your persistent queue system, compare the throughput times of your system without persistent queues enabled and then introduce persistent queue and measure the difference.
- **Memory:** When your system experiences sudden increases in data, your Workers need memory to store that data in temporary buffers. Consider what should happen in your system when memory is fully utilized.
- **Data storage:** When you back up data to a disk, that requires available disk space to use. Allocating memory and disk space can potentially be expensive. Be aware of the rate at which you consume disk memory and the general costs you estimate you will incur to write data to disk. Have systems in place for knowing when disk space is running out and adding more memory as needed.
- **Data throughput:** You need to understand your system's throughput under normal conditions so that you can identify potential bottlenecks in the system. Use the system monitoring tools to develop this baseline.

Key Decisions When Configuring Source Persistent Queues

You configure Source persistent queue behavior when you configure each individual Source. Consult the documentation for that Source for more information about initial configuration.

This section explains the decision-making process you should consider when implementing persistent queue on a Source. It explains the benefits and tradeoffs of different configuration settings for most use cases.

Whether to Enable Persistent Queues for Backpressure Events

The first decision you need to make is whether to use persistent queue to respond to backpressure. See When to Use Persistent Queues for a detailed explanation of which kinds of environments benefits from persistent queues.

In general, you should not use persistent queue is you are already using a backpressure mechanism on a upstream sender. Implementing backpressure mechanisms such as persistent queue on both Cribl Edge and the Source is redundant and could cause unnecessary data latency or impact data throughput.

The alternatives to using persistent queue are to block data or to drop events when a backpressure event occurs. Consider blocking events if you need Cribl to return block signals back to an upstream sender. If the sender also supports backpressure response mechanisms, the sender may stop sending data.

Consult the documentation for the sender to ensure you know how that specific sender handles a backpressure signal. In general, TCP-based senders support backpressure and stop sending data once Cribl Edge stops sending TCP acknowledgments back to it.

When and How to Store Data to Disk During a Backpressure Event

Once the backpressure queue behavior engages, you need to determine how it should write data to disk. Source persistent queues have two available options (modes) for storing data during a backpressure event:

Mode	Description
Always On	Uses persistent queue as a disk buffer for all events.
Smart	Only engages persistent queue upon backpressure from downstream receivers or Cribl internal processes.

The default is **Always On** mode. This mode is helpful if your business has a low tolerance for data loss. However, using **Always On** this mode could have impacts on the persistent queue optimization metrics, such as disk space, throughput, data latency, and CPU utilization.

Considerations for using **Always On** mode:

- This mode can increase memory demands compared to Sources without persistent queue enabled. If memory is a constraint, you can experiment with tuning the **Buffer size limit** setting. However, smaller

buffers will reduce the efficiency of disk writes, adding more latency.

- **Always On** mode has higher disk requirements than **Smart** mode. Cribl generally recommends that you allocate each Worker Node enough disk space for at least 1 day's worth of queued throughput. For example: If one Source is sending 1 TB/day to two evenly distributed Worker Processes, you should provision at least 500 GB queue storage for each Worker Process. Expand this basic calculation for additional Sources. From there, expand or contract it based on the typical length of outages you experience.
- Cribl also recommends that you deploy the highest-performance storage available. This prevents IOPS (input/output operations per second) and throughput bottlenecks for both read and write operations.

Considerations for using **Smart** mode:

- `Smart` mode only engages when necessary, such as when a downstream Destination becomes blocked *and* the **Buffer size limit** reaches its limit. When persistent queue is set to `Smart` mode, Cribl attempts to flush the queue when every new event arrives. The only time events stay in the buffer is when a downstream Destination becomes blocked.
- If you're using **Smart** mode and observe that it causes Source persistent queue to engage frequently, on option is to reduce the **File size limit** setting. Source persistent queue excessively triggers backpressure only when it reaches the last queued file *and* that file is large. Keeping this threshold low prevents that condition.

Consider experimenting with these modes to determine what is appropriate for your needs. Establish a baseline for system performance before you enable **Always On** and then observe the impacts when you turn that mode on.

For Cribl.Cloud Only: `Always On` is the Only Available Mode

On Cribl-managed Cribl.Cloud Workers with an Enterprise plan, you can enable persistent queues for supported Sources. By default, Source persistent queue is not enabled, so you must explicitly decide to turn it on.

When you enable Source persistent queue in Cribl.Cloud, the only available mode is `Always On`, which writes data to disk at all times. `Smart` mode is not available.

When you enable Source persistent queue in Cribl.Cloud, Cribl automatically allocates up to 1 GB of disk space per Source, per Worker Process. You cannot change this limit. This 1 GB limit applies to outbound uncompressed data, and the queue does not perform any compression. If the queue fills up, Cribl Edge blocks data flow, including outbound data.

When to Report Backpressure to the Sender

You can use the **Buffer size limit** setting to determine the maximum number of events to hold in memory before reporting backpressure to the sender. Once the number of events reaches this limit, Cribl Edge begins

writing the queue to disk. This setting defaults to `1000`.

This buffer is for all connections, not just per Worker Process. For that reason, this can dramatically expand memory usage. Connections share this limit, which may result in slightly lower throughput for higher number of connections. For higher numbers of connections, consider increasing the limit.

How Many Events to Commit During a Backpressure Event

Use the **Commit frequency** setting to determine how many events to send downstream before committing that Cribl Stream has read them. The defaults is `42`.

How Much Data to Store in Memory

When you enable persistent queues, you can configure the **File size limit** setting to specify the maximum amount of data stored in each queue file before the file closes and starts a new file. You can enter a value in units such as KB, MB, and more. If left unspecified, Cribl Stream/Edge defaults to 1 MB.

Additionally, you can use the **Queue size limit** setting to determine the maximum amount of disk space that Cribl allows the queue to consume on each Worker Process. Once the queue reaches this limit, the Source will stop queueing data and engage the configured backpressure mode. You can enter a value in units such as KB, MB, GB, or TB.

If the **Queue size limit** is not specified, it defaults to 5 GB. You can adjust this value to better suit your needs. For Sources with undefined **Queue size limit** values, queues exceeding 5 GB will trigger the configured queue-full behavior until you resize the limit.

To set a higher maximum value, navigate to **Worker Group/Fleet Settings** > **General Settings** > **Limits** > **Storage** > **Worker Process PQ size limit**. The corresponding key in the `limits.yml` file is `maxPQSize`. (See Config Files: limits.yml for more information.) The default for this setting is 1 TB. Before increasing this limit, consult Cribl Support for guidance.

> The information in this section applies only to on-prem or hybrid Worker Groups. For Cribl-managed Cribl.Cloud Organizations using persistent queue, Cribl automatically allocates up to 1 GB of disk space per Source, per Worker Process. You cannot change this limit.
>
> This 1 GB limit applies to outbound uncompressed data, and the queue does not perform any compression. If the queue fills up, Cribl Stream/Edge blocks data flow, including outbound data.

Whether to Compress Data

When the queue file reaches its maximum file size, you can use the optional **Compression** setting to apply compression to the data before the queue file closes. Currently, the only available compression codec is Gzip.

If you enable **Compression** and also enter a **Queue size limit**, be aware of how these options interact:

- Cribl Stream/Edge currently applies the **Queue size limit** against the **uncompressed** data volume entering the queue.
- With this combination, Cribl recommends that you set the **Queue size limit higher** than the volume's total available disk space, not including compression. This will maximize queue saturation and minimize data loss.

Where to Store Data and How Much Disk Space to Allocate

Use the **Queue file path** setting to determine the location for the persistent queue files. The default is `$CRIBL_HOME/state/queues`. When writing data, Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this path.

For queuing to operate properly, you must provide sufficient disk space. In Distributed mode, this minimum disk space configuration is set on each Worker Group/Fleet. You can access this setting at **Group/Fleet settings** > **General Settings** > **Limits** > **Min free disk space**.

If available disk space falls below this minimum threshold, Cribl Stream/Edge will stop maintaining persistent queues, and data loss will begin. The default minimum is 5 GB.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this value, Cribl Stream/Edge will append `/<worker-id>/<output-id>`.

Source Persistent Queue Notifications

Some Cribl licenses allow you to configure Notifications for Source persistent queues for on-prem deployments. In general, Cribl recommends setting Notifications for these events when using persistent queue to ensure you are aware of important system events. These Notifications always appear in Cribl Edge's user interface and internal logs. You can also send them to external systems.

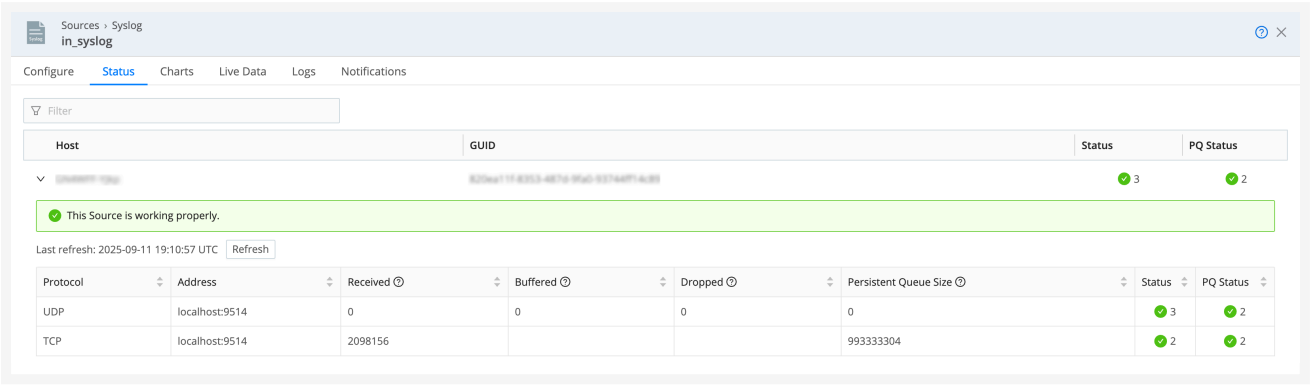
You can set Notifications for these Source persistent queue scenarios:

- Persistent queue files exceed a configurable percentage of allocated storage.
- A Source reaches the queue-full state, meaning it is running out of available disk space.
- Persistent Queue Usage (saturation).

See About Notifications for information about adding Notifications for Sources.

Monitor Source Persistent Queues

Cribl Edge provides metrics and health indicators to monitor your Source persistent queue through the **Status** tab on your Source. The **Status** tab aggregates metrics and health from all Worker Processes, allowing you to assess the health of your persistent queues and the status of the Source in real time.



Example of the Source Status tab

To view the Source persistent queue metrics:

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **More**, select **Sources**.
2. From the sidebar, select the type of Source you want to view.
3. Select the Source from the list of currently configured Sources.
4. In the resulting modal, select the **Status** tab.

Inside this view, you can expand individual Worker Nodes for more details.

For additional analysis, you can also use Monitoring dashboards:

1. Navigate to **Monitoring** on the product sidebar.
2. Navigate to **System** > **Queues (Sources)** and view relevant data.

Reference: Source Persistent Queue Fields

In the **Persistent Queue Settings** tab, you can optionally specify persistent queue storage, using the following controls. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues
- Optimize Source Persistent Queues (sPQ)

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the **Enable persistent queue** toggle. If enabled, PQ is automatically configured in `Always On` mode, with a maximum queue

> size of 1 GB disk space allocated per PQ-enabled Source, per Worker Process.
>
> The 1 GB limit is on uncompressed inbound data, and the queue does not perform any compression. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable persistent queue: Default is toggled off. When toggled on:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to the Cribl Stream data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from the Cribl Stream data processing engine.



`Smart` mode only engages when necessary, such as when a downstream Destination becomes blocked *and* the **Buffer size limit** reaches its limit. When persistent queue is set to `Smart` mode, Cribl attempts to flush the queue when every new event arrives. The only time events stay in the buffer is when a downstream Destination becomes blocked.

Buffer size limit: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. This buffer is for all connections, not just per Worker Process. For that reason, this can dramatically expand memory usage. Connections share this limit, which may result in slightly lower throughput for higher numbers of connections. For higher numbers of connections, consider increasing the limit.

Commit frequency: The number of events to send downstream before committing that Stream has read them. Defaults to `42`.

File size limit: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, and so forth. If not specified, Cribl Stream applies the default `1 MB`.

Queue size limit: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, and so forth. Can be set as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** in Group or Fleet settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file closes. Defaults to `None`; `Gzip` is also available.

In Cribl Stream 4.1 and newer, the Source persistent queue default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Optimize Source Persistent Queues (sPQ).

You can optimize Workers' startup connections and CPU load at **Group/Fleet settings** > Worker Processes.

12.4.3. Optimize Destination Persistent Queues

Cribl Edge's Destination persistent queues supplement in-memory queues to prevent data loss during a backpressure event, such as when a downstream Destination receiver is unreachable or slow to respond. See Supported Destinations for more information about which Destinations support persistent queueing.

Read this guide to learn how to optimize your Destination persistent queue. This guide is helpful when you are in the initial planning and configuration stage and also when you have an active deployment that you want to performance tune. This guide explains:

- How Destination persistent queues respond to a backpressure event.
- The overall goals and key metrics involved in optimizing Destination persistent queues.
- An in-depth discussion of the Destination persistent queue configuration options and their potential impact on your system.
- The available notifications, monitoring dashboards, and troubleshooting tools for persistent queues.

See also:

- About Persistent Queues
- About Destination Backpressure Triggers

On Cribl-managed Cribl.Cloud Workers with an Enterprise plan, you can enable persistent queue for supported Destinations. However, be aware that mode options and storage limits have specific restrictions in Cribl.Cloud. See For Cribl.Cloud Only: Whether to Use a Hybrid Group for more information.

How Destination Persistent Queues Work

Backpressure occurs when there is a temporary imbalance between the rates of inbound and outbound data that is either coming in or going out of Cribl Edge.

Different events trigger persistent queues on Destinations. For Destinations, the triggers depend on what kind of Destination it is, such as whether it is filesystem-based, TCP, UDP, and so on. See Destination Backpressure Triggers for more detailed information.

When persistent queue (PQ) engages, it writes and stores data on a local file system in your environment until the backpressure event resolves.

Destinations handle data rate imbalances using the following order of operations:

1. **In-memory buffers store data under normal conditions.** Under normal conditions, Destinations store data in in-memory buffers before sending the data to a receiver. These in-memory buffers on Destinations help absorb temporary imbalances between inbound and outbound data rates.
2. **If inbound data rates begin to spike, data shifts to storage in all the available buffers for that Destination.** Cribl Edge then attempts to output events at a more appropriate rate for the downstream receiver.



The threshold for triggering a backpressure event varies by Destination type. See Destination Backpressure Triggers for more information.

3. **A backpressure event triggers when buffers are full.**
4. **Backpressure behavior engages.** During a backpressure event, Cribl Edge responds in one of the following ways, based on the Destination **Backpressure behavior** options you configured on the Destination:

Setting	Description
Block	All data is blocked until the backpressure event has resolved. If set to Block, the output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender. If the sender also supports backpressure backpressure response mechanisms, the sender may stop sending data. Consult the documentation for the sender to ensure you know how that specific sender handles a backpressure signal. In general, TCP-based senders support backpressure and stop sending data once Cribl Edge stops sending TCP acknowledgments back to it.
Drop Events	The Destination will discard all new events until the backpressure event has resolved and the receiver is ready.
Persistent Queue	If the Destination supports it, the Destination will engage the persistent queue behavior.

5. **If enabled, persistent queue engages.** If the persistent queue behavior engages, Cribl Edge responds in one of the following ways, based on its configured backpressure mode:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

You can configure additional settings to determine:

- How much data to store in memory.
- What happens to data when the queue reaches its capacity limit.

6. **When the backpressure event ends, data starts flowing out of the persistent queue and into the Destination.** Cribl Edge begins to drain persistent queues when the receiver can accept data. In the case of load-balanced receivers, this means when at least one receiver can accept data. By default, after data flow is re-established, the persistent queue flushes every 10 seconds. It uses FIFO (first in, first out) event forwarding, ensuring Cribl Edge sends earlier queued events first when receivers recover. However, in Cribl Edge 4.1 and newer, you can instead prioritize new events by disabling **Strict ordering** on a Destination.

If you turn off Destination persistent queue (or shut down the whole Destination) before its queues drain, you will orphan any events still queued on disk. See Steps to Safely Disable and Clear Persistent Queues for more information.

Configure Destination Persistent Queue

You configure Destination persistent queue behavior when you configure each individual Destination. Consult the documentation for that Destination for more information about initial configuration.

This section provides a configuration quick start for experienced users. However, new Cribl Edge users should consider reading the full guidance in this topic to get a deeper understanding of the decisions and ideal system architecture design for persistent queue.

To configure persistent queue on a Destination:

1. In the **General Settings** tab, set the **Backpressure Behavior** option to **Persistent Queue**. See Whether to Enable Persistent Queues for Backpressure Events for guidance about the decision to enable persistent queue.
2. In the **Persistent Queue Settings** tab, select an appropriate **Mode** from the available options. See When and How to Store Data to Disk During a Backpressure Event for information about which mode to select.
3. Set the **File size limit** to specify the maximum amount of data stored in each queue file before the file closes and starts a new file. Set the **Queue size limit** to determine the maximum amount of disk space that Cribl allows the queue to consume on each Worker Process. See How Much Data to Store in Memory for more information about these settings.
4. Set the **Queue file path** to determine the location for the persistent queue files. See Where to Store Data and How Much Disk Space to Allocate for more information.

5. Optional: Set the **Compression** type to apply compression to the data before the queue file closes. See Whether to Compress Data for more information.
6. Set the **Queue-full behavior** to determine what happens when the persistent queue reaches its **Queue size limit** and runs out of space. See What Happens to Data when the Queue Reaches Its Configured Limits for more information.
7. The **Backpressure duration limit** setting displays when the **Mode** is set to `Backpressure`. It controls the duration a Destination must experience backpressure before data is queued to disk. The default waiting time is `30` seconds. Lowering this limit strengthens data loss prevention, while increasing it helps prevent unnecessary queue activation in environments with frequent, short network interruptions.
8. Optional: If needed, turn off **Strict Ordering** to more finely tune the **Drain rate limit (EPS)**. See How to Drain Persistent Queues When Backpressure Events End for more information.

Overview of Destination Persistent Queue Optimization

A well-optimized data throughput system can efficiently and reliably processes and transfer data from Sources to Destinations. Optimized systems maximize performance while minimizing latency, errors, and resource consumption. An optimized system has these characteristics:

- **High efficiency:** Data flows seamlessly at the maximum achievable rate without bottlenecks or excessive resource usage. The system gracefully handles backpressure events, ensuring that data intelligently throttles or queues without overwhelming downstream components.
- **Reliability:** The system consistently handles variations in data volume or speed without losing or corrupting data, even during spikes or failures.
- **Scalability:** The system adapts smoothly to increasing data loads or additional data senders or receivers without degradation in performance.
- **Low latency:** The system processes and delivers data in near real-time, meeting the required time constraints for your business use cases.
- **Cost-efficient resource allocation:** The system uses CPU, memory, disk, and network resources effectively, avoiding over-provisioning or waste.
- **Security and compliance:** The system transmits data securely, meeting all relevant privacy, encryption, and regulatory requirements.

Ultimately, you and your business stakeholders should have confidence in the system's performance, reliability, and ability to adapt to future needs. Persistent queue is an important tool to achieve this confidence.

Persistent Queue Optimization Metrics

The most effective way to performance tune is to understand which key metrics give insights into your persistent queues system's overall health. Effective system administrators study their deployments to establish

a performance baseline that they can use for comparison if a performance issue arises. Most system administrators understand their system well and can easily identify peak data periods.

Pay attention to these metrics:

- **CPU utilization:** When your system gets sudden increases in data loads, you might notice that your Worker Processes operate at full capacity and use all their CPU. Generally, that's a sign of inefficiency or the need for more resources. If needed, you can teleport into a single Worker and view all the processes and their individual CPU utilization. To see your CPU utilization, you can use the system monitoring tools to analyze CPU utilization. If needed, you can also teleport into a single Worker and view all the processes and their individual CPU utilization.
- **Data latency:** Some persistent queues can add latency because of the amount of overhead time it takes to write data to disk. As a starting point, assume that each event incurs a few milliseconds of added latency. Be aware that data latency can vary by file system, such as whether you are using a local file system or a Network File System (NFS). NFS file systems tend to introduce higher and more variable latency. To measure the latency impact of your persistent queue system, compare the throughput times of your system without persistent queues enabled and then introduce persistent queue and measure the difference.
- **Memory:** When your system experiences sudden increases in data, your Workers need memory to store that data in temporary buffers. Consider what should happen in your system when memory is fully utilized.
- **Data storage:** When you back up data to a disk, that requires available disk space to use. Allocating memory and disk space can potentially be expensive. Be aware of the rate at which you consume disk memory and the general costs you estimate you will incur to write data to disk. Have systems in place for knowing when disk space is running out and adding more memory as needed.
- **Data throughput:** You need to understand your system's throughput under normal conditions so that you can identify potential bottlenecks in the system. Use the system monitoring tools to develop this baseline.

Key Decisions When Configuring Destination Persistent Queues

You configure Destination persistent queue behavior when you configure each individual Destination. Consult the documentation for that Destination for more information about initial configuration.

This section explains the decision-making process you should consider when implementing persistent queue on a Destination. It explains the benefits and tradeoffs of different configuration settings for most use cases.

Whether to Enable Persistent Queues for Backpressure Events

The first decision you need to make is whether to use persistent queue to respond to backpressure. See When to Use Persistent Queues for a detailed explanation of which kinds of environments benefits from persistent queues.

In general, you should not use persistent queue is you are already using a backpressure mechanism on a downstream receiver. Implementing backpressure mechanisms such as persistent queue on both Cribl Edge and the Destination is redundant and could cause unnecessary data latency or impact data throughput.

The alternatives to using persistent queue are to block data or to drop events when a backpressure event occurs. Consider blocking events if you need Cribl to return block signals back to an upstream sender. If the sender also supports backpressure backpressure response mechanisms, the sender may stop sending data.

Consult the documentation for the sender to ensure you know how that specific sender handles a backpressure signal. In general, TCP-based senders support backpressure and stop sending data once Cribl Edge stops sending TCP acknowledgments back to it.

For Cribl.Cloud Only: Whether to Use a Hybrid Group

On Cribl-managed Cribl.Cloud Workers (with an Enterprise plan), persistent queue does not come with many of the configuration options described in the rest of this page. For Cribl.Cloud, enabling persistent queue only exposes the destructive **Clear Persistent Queue** button. Use this button with caution to avoid data loss. See Steps to Safely Disable and Clear Persistent Queues for more information.

When you enable persistent queues in Cribl.Cloud, Cribl automatically allocates up to 1 GB of disk space per Destination, per Worker Process. This 1 GB limit applies to outbound uncompressed data, and the queue does not perform any compression. If the queue fills up, Cribl Edge blocks data flow.

You cannot change this limit. If the queue fills up, Cribl Stream/Edge will block outbound data. Use a hybrid Group if you need to configure:

- Queue size
- Compression
- Backpressure modes
- Other persistent queue settings

When and How to Store Data to Disk During a Backpressure Event

Once the backpressure queue behavior engages, you need to determine how it should write data to disk. You have three available options (modes) for storing data during a backpressure event:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.

Mode	Description
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

Consider using the **Always On** mode if your business has a low tolerance for data loss. However, do so with some caution. Enabling this mode could have impacts on the persistent queue optimization metrics, such as disk space, throughput, data latency, and CPU utilization.

Consider experimenting with this mode to see whether it's right for your business needs. Establish a baseline for system performance before you enable **Always On** and then observe the impacts when you turn that mode on. Ensure you have enough disk storage space to handle your data storage needs.

How Much Data to Store in Memory

When you enable persistent queues, you can configure the **File size limit** setting to specify the maximum amount of data stored in each queue file before the file closes and starts a new file. You can enter a value in units such as KB, MB, and more. If left unspecified, Cribl Stream/Edge defaults to 1 MB.

Additionally, you can use the **Queue size limit** setting to determine the maximum amount of disk space that Cribl allows the queue to consume on each Worker Process. Once the queue reaches this limit, the Source will stop queueing data and engage the configured backpressure mode. You can enter a value in units such as KB, MB, GB, or TB.

If the **Queue size limit** is not specified, it defaults to 5 GB. You can adjust this value to better suit your needs. For Sources with undefined **Queue size limit** values, queues exceeding 5 GB will trigger the configured queue-full behavior until you resize the limit.

To set a higher maximum value, navigate to **Worker Group/Fleet Settings** > **General Settings** > **Limits** > **Storage** > **Worker Process PQ size limit**. The corresponding key in the `limits.yml` file is `maxPQSize`. (See Config Files: limits.yml for more information.) The default for this setting is 1 TB. Before increasing this limit, consult Cribl Support for guidance.

> The information in this section applies only to on-prem or hybrid Worker Groups. For Cribl-managed Cribl.Cloud Organizations using persistent queue, Cribl automatically allocates up to 1 GB of disk space per Source, per Worker Process. You cannot change this limit.
>
> This 1 GB limit applies to outbound uncompressed data, and the queue does not perform any compression. If the queue fills up, Cribl Stream/Edge blocks data flow, including outbound data.

Whether to Compress Data

When the queue file reaches its maximum file size, you can use the optional **Compression** setting to apply compression to the data before the queue file closes. Currently, the only available compression codec is Gzip.

If you enable **Compression** and also enter a **Queue size limit**, be aware of how these options interact:

- Cribl Stream/Edge currently applies the **Queue size limit** against the **uncompressed** data volume entering the queue.
- With this combination, Cribl recommends that you set the **Queue size limit higher** than the volume's total available disk space, not including compression. This will maximize queue saturation and minimize data loss.

Where to Store Data and How Much Disk Space to Allocate

Use the **Queue file path** setting to determine the location for the persistent queue files. The default is `$CRIBL_HOME/state/queues`. When writing data, Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this path.

For queuing to operate properly, you must provide sufficient disk space. In Distributed mode, this minimum disk space configuration is set on each Worker Group/Fleet. You can access this setting at **Group/Fleet settings** > **General Settings** > **Limits** > **Min free disk space**.

If available disk space falls below this minimum threshold, Cribl Stream/Edge will stop maintaining persistent queues, and data loss will begin. The default minimum is 5 GB.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this value, Cribl Stream/Edge will append `/<worker-id>/<output-id>`.

What Happens to Data when the Queue Reaches Its Configured Limits

You can use the **Queue-Full Behavior** setting to determine what happens when the persistent queue reaches its **Queue size limit** and runs out of space. There are two options:

Setting	Description
Block	Blocks all data until the backpressure event has resolved. If set to Block, the output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
Drop new data	The Destination will discard all new events until the backpressure event has resolved and the receiver is ready.

How to Drain Persistent Queues When Backpressure Events End

When a backpressure event ends, the persistent queue begins to drain to the downstream Destination. You can configure the order of event delivery and the rate at which Cribl Edge forwards events.

Use the **Strict Ordering** setting to configure the order that Cribl Edge sends data to the Destination. Toggle **Strict Ordering** on (default) to use FIFO (first in, first out) processing. This means Cribl prioritizes earlier queued events before sending newly arrived data. Toggle **Strict Ordering** off to change this behavior, prioritizing new data over the events already in the queue.

If you toggle **Strict Ordering** off, you must configure the **Drain rate limit (EPS)**. This setting allows you to throttle the rate at which events drain from the persistent queue to the Destination. The default value of `0` disables throttling. Throttling the drain rate helps Cribl manage the incoming stream of live data more effectively, preventing delays caused by queue processing. It also improves the throughput of active connections by reserving additional resources for them.

The Drain rate limit (EPS) is set in Events Per Second (EPS). To determine your system's EPS value:

1. Open the Monitoring page. From Cribl Edge's sidebar, select **Monitoring** to access dashboards displaying data for all Fleets and all nodes.
2. Filter by a specific node. Use the **Groups** menu at the top to restrict monitoring data to a single Edge Node.
3. Adjust the time granularity. Use the time menu change the display granularity from the default `15 min` to `1 day` for better accuracy.
4. Locate the EPS metrics. On the Monitoring page, find the **Events In and Out** metrics group. Copy the value listed under THROUGHPUT IN (AVG). For example, a typical value might be `31,000 EPS`.

Cribl recommends setting the **Drain Rate Limit** to 10% of the Worker Process EPS to prioritize live data processing. To calculate an appropriate drain rate:

1. Determine EPS per Worker Process. Divide the Worker Process EPS by the number of Worker Processes (WPs). For example: `31,000 EPS / 14 WPs = 2,200 EPS per WP`.
2. Calculate 10% of EPS per Worker Process. Multiply the result by `0.10` to find the recommended drain rate. For example: `10% of 2,200 EPS = 220 EPS`.
3. Configure the **Drain Rate Limit**. In the persistent queue settings for the Destination, enter this value in the **Drain rate limit (EPS)** field. For example: `220`.

After applying this setting, monitor the queue's drain rate. If the queue is draining too slowly, carefully increase the **Drain rate limit (EPS)** while ensuring it doesn't interfere with live data processing. Avoid setting the limit too high, as this can impact the performance of live streams. If needed, consider optimizing Worker startup connections and managing CPU load to further enhance performance at **Fleet Settings** > Worker Processes.

Cribl's guarantee for data delivery is that all data is delivered at least once, but not exactly once. Sometimes Cribl Edge sends duplicate data events when the persistent queue drains.

Steps to Safely Disable and Clear Persistent Queues

If you want to disable persistent queuing, always wait for the queue to fully drain. The **Clear Persistent Queue** button is available as a destructive fallback. This action frees up disk space, but it also deletes the queued data *without* sending it on to receivers.

After the queue has drained, then you can select a different backpressure behavior. Save the configuration after changing it.

Destination Persistent Queue Notifications

Some Cribl licenses allow you to configure Notifications for Destination persistent queues for on-prem deployments. In general, Cribl recommends setting Notifications for these events when using persistent queue to ensure you are aware of important system events. These Notifications always appear in Cribl Edge's user interface and internal logs. You can also send them to external systems.

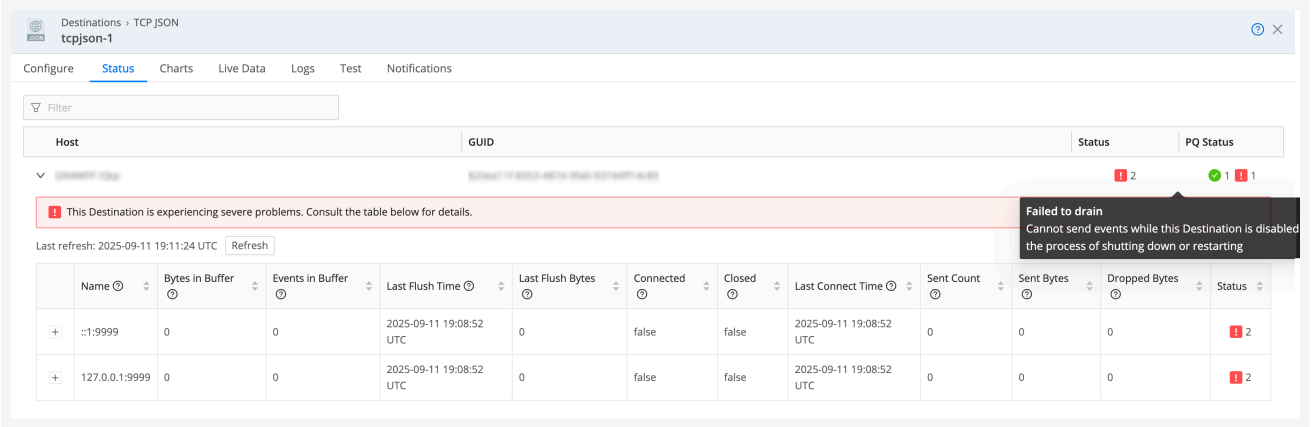
You can set notifications for these scenarios:

- Persistent queue files exceed a configurable percentage of allocated storage.
- A Destination reaches the queue-full state, meaning it is running out of available disk space.
- A Destination backpressure engages, causing blocked or dropped events.
- Persistent Queue Usage (saturation).

See About Notifications for information about adding Notifications for Destinations.

Monitor Destination Persistent Queues

Cribl Edge provides metrics and health indicators to monitor your Destination persistent queue through the **Status** tab on your Destination. Cribl Edge provides metrics and health indicators to monitor your Source persistent queue through the **Status** tab on your Source. The **Status** tab aggregates metrics and health from all Worker Processes, allowing you to assess the health of your persistent queues and the status of the Destination in real time.



Example of the Destination Status tab

To view the Destination persistent queue metrics:

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **More**, select **Destinations**.
2. From the sidebar, select the type of Destination you want to view.
3. Select the Destination from the list of currently configured Destination.
4. In the resulting modal, select the **Status** tab.

Inside this view, you can expand individual Worker Nodes for more details.

For additional analysis, you can also use Monitoring dashboards:

1. Navigate to **Monitoring** on the product sidebar.
2. Navigate to **System** > **Queues (Sources)** and view relevant data.

Troubleshooting Tools

Cribl internal logs for Cribl-managed Workers might include entries labeled `Revived PQ Worker Process`. These indicate where a Cribl-managed Worker was newly assigned to drain a persistent queue left over from a shut-down Worker Node. You can use these log events to troubleshoot Cloud PQ issues.

Reference: Destination Persistent Queue Fields

The **Persistent Queue Settings** tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**. Persistent queue buffers and preserves incoming events when a downstream Destination has an outage or experiences backpressure.

Before enabling persistent queue, learn more about persistent queue behavior and how to optimize it with your system:

- About Persistent Queues

- Optimize Destination Persistent Queues (dPQ)
- Destination Backpressure Triggers

On Cribl-managed Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described at the end of this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream/Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Use this menu to select when Cribl Stream/Edge engages the persistent queue in response to backpressure events from this Destination. The options are:

Mode	Description
Error	Queues and stores data on a disk when the Destination is unavailable or in an error state.
Backpressure	Queues and stores data to a disk when it detects backpressure from the Destination until the backpressure event resolves.
Always On	Cribl Stream or Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

If a Worker/Edge Node starts with an invalid **Mode** setting, it automatically switches to **Error** mode. This might happen if the Worker/Edge Node is running a version that does not support other modes (older than 4.9.0), or if it encounters a nonexistent value in YAML configuration files.

File size limit: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Queue size limit: The maximum amount of disk space that the queue can consume on each Worker Process. When the queue reaches this limit, the Destination stops queueing data and applies the **Queue-full** behavior. Defaults to `5 GB`. This field accepts positive numbers with units of `KB`, `MB`, `GB`, and so on. You can set it as high as `1 TB`, unless you've configured a different **Worker Process PQ size limit** on the Worker Group/Fleet Settings page.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream/Edge will append `/<worker-id>/<output-id>` to this value.

Compression: Set the codec to use when compressing the persisted data after closing a file. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue begins to exert backpressure. A queue begins to exert backpressure when the disk is low or at full capacity. This setting has two options:

- **Block**: The output will refuse to accept new data until the receiver is ready. The system will return block signals back to the sender.
- **Drop new data**: Discard all new events until the backpressure event has resolved and the receiver is ready.

Backpressure duration Limit: When **Mode** is set to `Backpressure`, this setting controls how long to wait during network slowdowns before activating queues. A shorter duration enhances critical data loss prevention, while a longer duration helps avoid unnecessary queue transitions in environments with frequent, brief network fluctuations. The default value is `30` seconds.

Strict ordering: Toggle on (default) to enable FIFO (first in, first out) event forwarding, ensuring Cribl Stream/Edge sends earlier queued events first when receivers recover. The persistent queue flushes every 10 seconds in this mode. Toggle off to prioritize new events over queued events, configure a custom drain rate for the queue, and display this option:

- **Drain rate limit (EPS)**: Optionally, set a throttling rate (in events per second) on writing from the queue to receivers. (The default `0` value disables throttling.) Throttling the queue drain rate can boost the throughput of new and active connections by reserving more resources for them. You can further optimize Worker startup connections and CPU load in the Worker Processes settings.

Clear Persistent Queue: For Cloud Enterprise only, click this button if you want to delete the files that are

12.4.4. About Destination Backpressure Triggers

This page describes the conditions that can trigger backpressure in Cribl Edge Destinations. Backpressure exists when a destination receives more data than it can currently send. It typically results from network connectivity problems or when a Destination is receiving data faster than it can forward that data to the downstream service.

When a Destination signals backpressure, it notifies the Sources generating data that its buffers are full and that the incoming data flow should pause until there is room for more data.

You can configure your desired behavior through a Destination's **Backpressure Behavior** drop-down. Where other options are not displayed, Cribl Edge's default behavior is **Block**. For details about all the above behaviors and options, see Persistent Queues.

HTTP-Based Destinations

HTTP-based Destinations trigger backpressure when they encounter errors while connecting or when they receive certain HTTP response codes. If the HTTP call returns a `500` series (`500-599`) HTTP code, the Destination adds the events in the buffer back to its internal buffer and tries to send them again. The retry attempt can also trigger backpressure, depending on the frequency at which this happens.

If the Destinaton receives a `400` series response code, it will drop (not retry) the events, because these response codes indicate an error in payload or formatting (for example). See this Mozilla page on the `400 Bad Request` code for more information.

Finally, HTTP-based Destinations trigger backpressure when their processing loads exceed the specified limit. HTTP-based Destinations handle requests by creating a connection, sending the request payload, and receiving and parsing a response. Because this process can take time, HTTP Destinations allow more than one request to be in progress at a time. When the number of requests in flight exceeds the value set in **Advanced Settings** > **Request concurrency**, the Destination will trigger backpressure.

The following conditions will trigger backpressure:

- Connection failure.
- Connection or request timeouts.
- Data overload, such as when the Source sends more data than the Destination will accept.
- Reaching the limit on concurrent (in-flight) requests.
- HTTP `500` series responses.

You can set the number of in-flight or concurrent requests in **Advanced Settings** > **Request concurrency**.

See Available Destinations for information about which Destinations are HTTP-based.

Filesystem-Based Destinations

Filesystem-based Destinations that send data to cloud-based services trigger backpressure when Cribl Edge can't move a file or upload it to the cloud.

Filesystem-based Destinations queue events in files on disk in a staging directory. When a batch of events is ready for transmission, Cribl Edge closes the file, optionally compresses it, and transmits the file to the downstream service.

For the Filesystem Destination, the batch simply moves from the staging directory to the output directory.

For other filesystem-based Destinations, Cribl Edge uploads the file to a cloud service, such as AWS S3, Azure Blob Storage, or other. For these Destinations, a failure to upload the file will trigger backpressure.

These scenarios include:

- Permissions problem writing to the staging directory, such as when the user running the Cribl Edge process does not have permission to write to the staging directory.
- The filesystem for the staging or output directory is full (impacts Filesystem Destination only).
- The credentials provided for the cloud service are invalid.
- The Destination doesn't have the permissions required to upload to the cloud service.
- Cloud service permissions issue, such as when an upload fails because the cloud user (identified in credentials) does not have write permissions.
- Network connectivity - The Destination can't communicate with the cloud service due to network connectivity issues.

Object Storage Destinations and Backpressure

Object-based Destinations, such as Amazon S3, Azure Blob Storage, Google Cloud Storage, and similar services, do not support Destination persistent queues. Instead, these Destinations use a local staging directory to buffer files before uploading them to cloud storage.

Resilience for object storage Destinations depends on staging disk capacity and upload success. Unlike streaming Destinations with persistent queues, there is no separate queue to absorb prolonged outages.

When configuring the **Backpressure behavior** setting for object storage Destinations, consider the impact on other Destinations receiving the same input data:

- If **Block** is selected and the Destination becomes unhealthy (for example, due to credential issues or network problems), it can stall delivery to all other Destinations that receive the same events.

- If **Drop** is selected, Cribl Edge discards events for the affected Destination while other Destinations continue to receive data.

For archival or less-critical paths, **Drop** may be preferable to **Block** to avoid stalling more important downstream Destinations. However, this means accepting potential data loss on the archive path during outages.

For sizing and configuration guidance, see S3 Better Practices.

See Available Destinations for information about which Destinations are filesystem-based.

TCP-Based Destinations

TCP Sources establish a persistent connection with a Destination and send data. These Sources maintain connection to the Destination and will re-establish that connection if they detect a close or write timeout.

If the connection fails, Cribl Edge uses a backoff algorithm to avoid spiking the CPU. This backoff logic makes the first retry at 2 seconds, then gradually increases the time between retries up to a maximum of 60 seconds.

The following conditions can trigger backpressure from a TCP-based Destination:

- Connection failure.
- Authentication failure.
- Write timeout, where a Source sends data but doesn't receive an `ack` at the TCP layer. In this case, Cribl Edge closes the connection to the Destination and tries to re-establish it.
- The Destination receives data from the Source faster than it can send data over the network to the downstream service, causing internal buffers to fill up. This condition can occur due to a slow network connection or slow processing on the downstream service.When space in the internal buffers becomes available, backpressure is relieved, allowing the Source to send more data.

See Available Destinations for information about which Destinations are TCP-based. Some Destinations use either TCP or UDP to send messages, but they trigger backpressure and use the backoff algorithm described above only when used with the TCP protocol.

Kafka

Kafka is a binary protocol that runs over TCP. The protocol defines three roles - producers, consumers, and brokers.

Producers send data to a Kafka topic (think of it as a message queue). Kafka consumers read data from Kafka topics. Brokers receive data from topic producers and send data to topic consumers, meaning they serve as go-betweens for producers and consumers.

Cribl Edge's Kafka-based Destinations are producers. They send data to Kafka brokers. Since the protocol is TCP-based, the conditions that trigger backpressure are similar to those of Cribl Edge's TCP Destinations.

The specifics for connection and request retries are defined by the Kafka JS library in conjunction with the configuration parameters defined for each Destination. In all cases, the Destinations have internal buffers that are sent to the Broker when they're considered full (on a size and time basis).

The following conditions can trigger backpressure for Kafka-based Sources:

- Failure to connect to the broker, lack of network connectivity, or the broker is down.
- Invalid credentials - Destination connects successfully, but the connection is rejected due to a failure to authenticate.
- Invalid permissions - The credentials assigned do not have proper permissions to send events to the topic.
- The Destination is attempting to send data faster than the broker can receive it. Backpressure will be relieved when internal buffers have room for more events.

Kafka Retries and PQ

In some situations, Kafka Destinations may take a long time to engage the PQ (1 to 6 minutes by default).

This delay can result from the default settings used with the Kafka Destination. The relevant settings are all in the **Advanced Settings** UI:

- **Connection timeout (ms)**: Defaults to `10000`,
- **Request timeout (ms)**: Defaults to `60000`.
- **Retry limit**: Defaults to `5`.

These default settings mean that Cribl Edge will retry five times after a failed connection, with a 10-second timeout between each attempt. It will also retry five files after a failed request, with a timeout of 60 seconds between each attempt.

For faster PQ engagement, update these values. You can use smaller values for **Connection timeout** and **Request timeout** to decrease the interval between retries. You can also use smaller values for **Retry limit** or even set it to `0` to eliminate retries altogether.

UDP

UDP is a connectionless protocol, so there is no concept of backpressure at the network layer. As a result, UDP Destinations will never trigger backpressure.

See Available Destinations for information about which Destinations support UDP. Except for SNMP Trap, these Destinations also support TCP, but the UDP versions do not trigger backpressure.

SQS

The SQS (Simple Queue Service) Destination uses the AWS SQS library to send events to the Destination's configured queue. Like Kafka, this Destination acts as a producer that publishes messages to the provisioned SQS queue.

This Destination has five buffers with 10 events per buffer for a total of 50 events that can be queued before backpressure is enabled. The reason that backpressure might be generated for this Destination include:

- Invalid credentials.
- Invalid permissions.
- Queue does not exist and **Create Queue** is disabled.
- Lack of network connectivity.

Amazon Kinesis

Kinesis runs over HTTPS and behaves like Pub/Sub Destinations.

This Destination serializes and publishes events in a newline-delimited JSON format. In the context of the Kinesis Destination, it behaves as a producer where a Kinesis source is considered a consumer. Since the protocol is HTTP-based, the conditions that trigger backpressure are similar to those of other Cribl Edge HTTP Destinations.

The Kinesis library defines the specifics for connection and request retries in conjunction with the configuration parameters defined for each Destination. In all cases, the Destinations have internal buffers that are sent to the Broker when they're considered full (on a size and time basis).

The following conditions can trigger backpressure for Kinesis-based Destinations:

- Failure to connect to the Kinesis service, network connectivity, or the service is down.
- Invalid credentials - Connects successfully but connection is rejected due to a failure to authenticate.
- Invalid permissions - The credentials assigned do not have proper permissions to send events to the Kinesis stream.
- The Destination is attempting to send data faster than the service can receive it. Backpressure will be relieved when internal buffers have room for more events.

Google Cloud Pub/Sub

Google Cloud Pub/Sub is similar to Kafka and Kinesis in that there are producers that send data and consumers that receive data. The Cribl Edge Google Cloud Pub/Sub Destination is a producer that sends data to the Google

Cloud Pub/Sub service. Google Cloud Pub/Sub leverages gRCP, however, which uses protocol buffers as both its interface definition language (IDL) and as its underlying message interchange format.

The conditions that trigger backpressure for a Google Cloud Pub/Sub Destination are:

- Failure to connect to the Google Pub/Sub service, network connectivity, or the service is down.
- Invalid credentials - Connects successfully but connection is rejected due to a failure to authenticate.
- Invalid permissions - The credentials assigned do not have proper permissions to send events to the topic.
- The Destination is attempting to send data faster than the service can receive it. Backpressure will be relieved when internal buffers have room for more events.

OpenTelemetry

OpenTelemetry supports gRPC or HTTP to send messages. For the gRPC protocol, the same rules described for other TCP Destinations generally apply to OpenTelemetry. The key difference is that this Destination uses the OpenTelemetry library, which handles write timeouts and reconnection logic.

For the HTTP protocol, the same rules apply to our HTTP Destinations for retries based on connectivity or `500` series response codes.

Load-Balanced Destinations

HTTP-Based Load-Balanced Destinations

Some HTTP-based Destinations support two modes - single host and multiple load-balanced hosts.

With a load-balanced version of an HTTP Destination, Cribl Edge spreads events across all configured Destinations, using a weighted algorithm. Under normal circumstances, this algorithm spreads the load across the configured Destinations according to the weight assigned to the Destination.

When one of the load-balanced Destinations is out of service (due to a write timeout or `500` series response code - buffers full), Cribl Edge removes that Destination from the list of hosts and moves any in-flight buffers to the next available host. Cribl Edge will try to resend events to the downed host in the background, and it will resume sending data to that host once the connection is restored.

See Available Destinations for information about which HTTP-based Destinations support load balancing.

TCP Load-Balanced Destinations

Some TCP-based Destinations support two modes - single host and multiple load-balanced hosts.

With a load-balanced TCP Destination, Cribl Edge spreads events across all configured Destinations using a weighted algorithm. Under normal circumstances, this algorithm spreads the load across the configured Destinations according to the weight assigned to each Destination.

When one of the load-balanced Destinations is out of service (due to a write timeout or connection failure), Cribl Edge removes that Destination from the list of hosts and moves any in-flight buffers to the next available host. Cribl Edge will try to reconnect to the downed host and will resume sending data to that host once the connection is restored.

TCP load-balanced Destinations can be configured with one or multiple receivers. If one or more receivers go down, Cribl Edge will continue sending data to any healthy receivers.

See Available Destinations for information about which TCP Destinations support load balancing.

Output Router

The Output Router is a wrapper destination that groups together 1..*N* Destinations. It's used in conjunction with event filters to route data to a selected Destination. The Output Router Destination emits backpressure only when one of its downstream Destinations is blocking.

External Load Balancers

External load balancers placed in front of Edge Nodes generally do not detect backpressure states within the data Pipeline. If a Edge Node still accepts network connections, the load balancer typically considers it healthy, even when event flow is blocked by downstream backpressure.

This means:

- Load balancing alone should not be treated as protection against blocked Destinations.
- A Edge Node under backpressure may continue to receive new data from the load balancer, potentially leading to dropped events if Source buffering is exhausted.
- Health checks based solely on TCP connectivity or HTTP endpoints may not reflect the Edge Node's ability to process and deliver events.

To detect backpressure conditions, use Cribl Edge's built-in Notifications for **Destination Backpressure Activated** or monitor Destination health directly.

12.4.5. Backpressure Impacts To Sources

When a Destination experiences backpressure, Cribl Edge must manage the incoming data flow to prevent overwhelming the system. If persistent queue (PQ) is not enabled, backpressure propagates back through the data Pipeline to the Sources, which can impact data flow in different ways depending on the Source type.

This page explains how backpressure affects Sources when PQ is not in use and the **Backpressure behavior** is set to **Block**. For information about backpressure behavior options, see Data Delivery to Unreachable Destinations.

How Backpressure Propagates Without PQ

When a Destination signals backpressure and PQ is not enabled, the Cribl Edge event processors stop accepting new data. This creates a chain reaction that propagates back through the Pipeline:

1. The Destination internal buffers fill up.
2. The event processors stop reading from Source buffers.
3. Source buffers fill up.
4. The Source signals backpressure to upstream senders (when supported).

The speed at which this propagation occurs depends on buffer sizes throughout the Pipeline. A brief backpressure event might not immediately affect all Sources, but sustained backpressure will eventually impact all connected senders. For Sources that support backpressure signaling, senders are blocked and data queues on the sender side. For Sources that do not support backpressure signaling (such as UDP-based Sources), data is dropped.



The Destination does not directly notify individual Sources about backpressure. Instead, for Sources that support backpressure signaling, backpressure propagates indirectly when the Cribl Edge event processors stop reading from Source buffers, causing those buffers to fill. The specific mechanism varies by Source type. See the sections below for details.

The following sections describe how this backpressure propagation affects different Source types when PQ is not enabled.

TCP-Based Sources

For TCP-based Sources (such as Splunk TCP, Syslog over TCP, TCP JSON, and TCP Raw), backpressure propagates through the following mechanism:

1. When the Destination buffers fill, the Cribl Edge event processor stops reading from the TCP socket buffer.
2. When the TCP socket buffer fills, the TCP/IP stack notifies the sender to stop sending data.
3. The sender pauses transmission until the socket buffer has capacity again.

This mechanism relies on TCP flow control, which is built into the protocol. When Cribl Edge stops acknowledging received data, the TCP stack on the sender automatically slows or pauses transmission. During this time, the sender's outbound data queues in its local TCP buffer. When Cribl Edge resumes reading from the socket (because the Destination has capacity again), the TCP stack automatically resumes transmission and the sender's buffer drains.

Per-Connection Behavior

When the TCP socket buffer fills, the TCP/IP stack signals the sender to stop sending data. Because this occurs at the individual socket level, some senders may experience blocking sooner than others, even if they are sending data to the same Destination.

This means:

- Different senders connected to the same Source may experience backpressure at different times.
- Senders transmitting higher volumes of data will fill their socket buffers faster and experience backpressure sooner.
- Senders with slower network connections may have smaller effective buffer capacities and experience backpressure sooner.



Only TCP-based Sources support this flow control mechanism. UDP-based Sources (such as UDP Raw and SNMP Trap) do not have built-in backpressure support because UDP is a connectionless protocol. For UDP Sources, Cribl Edge drops incoming data when buffers are full.

Pull-Based Sources

For pull-based Sources, Cribl Edge has built-in intelligence to detect backpressure and pause data retrieval. When backpressure is detected, pull-based Sources stop or slow their data collection activities.

The following pull-based Sources are affected:

Source Type	Backpressure Behavior
S3, Azure Blob Storage, and other object storage Sources	File downloads pause when backpressure is detected.

Source Type	Backpressure Behavior
Database Collectors	Results streaming pauses during backpressure.
REST Collectors	GET requests pause during backpressure.
Kafka, Confluent Cloud, Amazon MSK	Polling from Kafka-based Sources stops during backpressure.
Amazon Kinesis Data Streams	Stream reads pause during backpressure.
Amazon SQS, Google Cloud Pub/Sub	Message retrieval pauses during backpressure.

Side Effects of Pausing Pull-Based Sources

While pausing data collection prevents overwhelming the Pipeline, it can introduce side effects.

File Download Timeouts

Backpressure causes file downloads from object storage (such as S3 or Azure Blob Storage) to pause. If backpressure persists for too long, the download may time out. This can result in:

- **Duplicate data**: If Cribl Edge retries the download after a timeout, you may ingest the same data twice.
- **Data loss**: If the source data is purged or expires before Cribl Edge can recollect it, that data is lost.

To mitigate this risk, consider increasing the **Socket timeout** setting in **Advanced Settings** for the Source, or enable persistent queue to prevent prolonged backpressure events.

Database Resource Consumption

When Database Collectors pause results streaming due to backpressure, the database query remains active on the database server. This means:

- The database server continues to consume resources (CPU, memory, connections) for the paused query.
- Database connections remain held, which can exhaust the connection pool and block other database operations.
- The Database Collector runs on only one Worker Process at a time, so if that process is blocked, the entire Database Collector job run is affected.

For long-running database queries, monitor your database server resources and connection pool usage during backpressure events.

Kafka Consumer Group Rebalancing

When Kafka-based Sources pause polling for extended periods, the Kafka broker may consider the consumer inactive and trigger a consumer group rebalance. This can cause:

- Temporary disruption to other consumers in the same consumer group.
- Potential duplicate processing when the consumer rejoins.

Impact on Multi-Destination Routing

When a sender is blocked due to backpressure from one Destination, it affects data delivery to all Destinations receiving data from that sender, even Destinations that are not experiencing backpressure.

This occurs because event cloning for multi-Destination routing happens after data passes through the Routes. If a sender is blocked:

1. Data from that sender cannot flow through the Routes.
2. The Clone Function or Output Router cannot create copies for other Destinations.
3. All Destinations expecting data from that sender stop receiving data.

Example Scenario

Consider a configuration where:

- A Syslog Source sends data to two Destinations via an Output Router.
- Destination A experiences backpressure.
- Destination B is healthy and can accept data.

When Destination A blocks and the sender is affected:

- No new data from that sender reaches Destination B either.
- This occurs even though Destination B has capacity.
- The block persists until backpressure on Destination A resolves.



To prevent one Destination from blocking all data flow, set non-critical Destinations to **Drop** or enable persistent queue.

12.5. About Load Balancing

Cribl Edge will attempt to load-balance outbound data as fairly as possibly across all endpoints. For example, if FQDNs/hostnames are used as the Destination addresses, and each resolves to 5 (unique) IPs, then each Worker Process will have its # of outbound connections = {# of IPs x # of FQDNs} for purposes of the Destination.

Data is sent by all Worker Processes to a randomly selected endpoint, and the amount sent to each connection depends on these parameters:

1. Respective destination **weight**: If a **Load Weight** is set to `0`, Cribl Edge will not try to connect to that IP (nor to any IPs resolved for that FQDN). When a connection is blocked, Cribl Edge will apply a 10% penalty to the load weight, and will select the connection with the lower adjusted load.
2. Respective destination **historical data**: By default, historical data is tracked for 300s. Cribl Edge uses this data to influence the traffic sent to each destination, to ensure that differences decay over time, and that total ratios converge towards configured weights.

Sources share a single Worker Process for load balancing, so you don't need to dedicate a separate Worker Process for load balancing for each Source. Load balancing Worker Processes have a high throughput rate because they only handle Event Breakers.



On Destinations that support load balancing, Cribl recommends enabling the feature, even if you only have one hostname - because this hostname can expand to multiple IPs. If you must disable load balancing, then to ensure robust connections, enable the alternative **Advanced Settings** > **Round-robin DNS** option. This will allow Cribl Edge to follow the DNS host's IP rotation, preventing connection pinning.

If a request fails, Cribl Edge will resend the data to a different endpoint. Cribl Edge will block only if **all** endpoints are experiencing problems.

Example: Equal Distribution

Suppose we have two connections, A and B, each with weight of `1` (i.e., they are configured to receive equal amounts of data). Suppose further that the load-balance stats period is set at the default `300s` and - to make things easy - for each period, there are 200 events of equal size (bytes) that need to be balanced.

Interval	Time Range	Events to balance
1	*time=0s —> time=300s*	**200**

Both A and B start this interval with 0 historical stats each.

Let's assume that, due to various circumstances, 200 events are "balanced" as follows: `A = 120 events` and `B = 80 events` - a difference of **40 events** and a ratio of **1.5:1**.

Interval	Time Range	Events to balance
2	*time=300s —> time=600s*	**200**

At the beginning of interval 2, the load-balancing algorithm will look back to the previous interval stats and carry **half** of the receiving stats forward. In other words, connection A will start the interval with **60** and connection B with **40**. To determine how many events A and B will receive during this next interval, Cribl Edge will use their weights and their stats as follows:

Total number of events: `events to balance + stats carried forward = 200 + 60 + 40 = 300`.

Number of events per each destination (weighted): `300/2 = 150` (they're equal, due to equal weight).

Number of events to send to each destination `A: 150 - 60 = 90` and `B: 150 - 40 = 110`.

Totals at the end of interval 2: `A=120+90=210`, `B=80+110=190`, a difference of **20 events** and a ratio of **1.1:1**.

Over the subsequent intervals, the difference becomes exponentially less pronounced, and eventually insignificant. Thus, the load gets balanced fairly.

Example: Unequal Distribution

Suppose you have three connections, A, B, and C, weighted to receive unequal amounts of data:

- A: 1
- B: 2
- C: 7

Assume that the load-balance stats period is set at the default `300s` and for each period, 200 events of equal size (bytes) need to be balanced:

Interval	Time Range	Events to balance
1	*time=0s —> time=300s*	**200**

All three connections (A, B, and C) start this interval with 0 historical stats each.

Let's assume that 200 events are "balanced" as follows: `A = 30 events`, `B = 50 events`, and `C = 120 events` - a difference of **90 events** and a ratio of **3:5:12**.

Interval	Time Range	Events to balance
2	*time=600s —> time=600s*	**200**

At the beginning of interval 2, the load-balancing algorithm will look back to the previous interval stats and carry **half** of the receiving stats forward. This means that connection A will start the interval with **15**, connection B with **25**, and connection C with **60**. To determine how many events each connection will receive during this next interval, Cribl Edge will use their weights and their stats as follows:

Total number of events: `events to balance + stats carried forward = 200 + 15 + 25 + 60 = 300`.

Number of events per each destination (weighted): `300 / (1 + 2 + 7) = 300 / 10 = 30`.

Number of events to send to each destination `A: 30 - 15 = 15`; `B: 60 - 25 = 35`; `C: 210 - 60 = 150`.

Totals at the end of interval 2: `A: 30 + 15 = 45`; `B: 50 + 35 = 85`; `C: 120 + 150 = 270`.

The difference between the highest connection (C) and the lowest (A) is **225 events**. The ratio at the end of interval 2 is approximately **1:1.89:6**.

Over the subsequent intervals, the difference becomes exponentially less pronounced, and eventually insignificant. Thus, the load gets balanced fairly.

12.6. Transfer Data Between Workspaces Or Environments

Cribl Edge provides you with the ability to transfer data between Cribl Edge Nodes that report to the **same** Leader or different Leaders, via the Cribl TCP and Cribl HTTP Source and Destination pairs.

You can enable data sharing across Edge Nodes that do not share a Leader as long as both Organizations have **active, paid Cribl.Cloud plans** or **active, paid licenses**.

We designed this workflow to support scenarios where you can't or don't want to create a direct connection between Leader Nodes, such as:

Use case	Description
High/Low Security Environments	Isolated and protected network environments that handle sensitive data (for example, air-gapped networks managed by Cribl Edge) sending data to an external Cribl Stream processing tier.
Cribl Stream to Cribl Stream	Send data between Cribl Stream environments or Workspaces that exist within the overall Cribl system that you own and pay for.
Decentralized Management	Cribl Edge environments managed by separate Leaders forwarding data to a central Cribl Stream instance.

Key Aspects of Data Transfer Between Environments

Understand how you're billed for data transfers between environments and separate, paid Organizations.

Connectivity

The sending and receiving Organizations do not need to share a Leader. Use a paired **Cribl HTTP** or **Cribl TCP** Source/Destination.

Trust and Billing

- The **single ingest** benefit applies only when Workspaces are in the same **Cribl.Cloud Organization**, or when on-prem Leaders have the exact same paid license key installed.
- You can transfer data between separate paid **Cribl.Cloud Organizations**, but the receiving Organization will incur ingest charges.



Example: Customer to MSSP across separate Cloud Organizations

> An enterprise customer (paid Cribl.Cloud Organization A) can send data via **Cribl HTTP** or **Cribl TCP** to their MSSP (paid Cribl.Cloud Organization B). This setup is supported. Because the Organizations are different, the **single ingest** benefit does not apply across them; the MSSP Organization will be billed for the received data.

Understand Environment Compatibility

Sender → Receiver	Org/Workspace scenario	Cribl HTTP/TCP Support	Receiver Billed for Ingest?	Example Deployments	Notes
Same Leader	Same Workspace		No	• On-prem Worker Group A → Worker Group B under the same on-prem Leader. • Cribl.Cloud hybrid Worker Group A → Cribl.Cloud managed Worker Group B in the same Workspace. • Cribl Edge Fleet → Cribl Stream Worker Group.	This is the default Cribl to Cribl pattern.
Different Leaders	Same on-prem license key		No	• Two on-prem Stream deployments sharing a license key. • Edge Fleet under Leader 1 → Stream Worker Group under Leader 2, same key.	Treated as a single paid environment via shared key.

Sender → Receiver	Org/Workspace scenario	Cribl HTTP/TCP Support	Receiver Billed for Ingest?	Example Deployments	Notes
Different Leaders	Two Workspaces in the same Cribl.Cloud Org		No	• Cribl.Cloud-managed hybrid Worker Group in Workspace A → Cribl.Cloud-managed Worker Group in Workspace B. • Edge → Stream across Workspaces in the same Org.	Requires supported versions and shared Org – uses auth tokens on 4.15+.
Different Leaders	Different Cribl.Cloud Orgs		Yes	Enterprise customer Org A → MSSP Org B via Cribl HTTP/TCP.	Supported, but the receiving Org is billed. No single-ingest across Orgs.
On-prem → Cribl.Cloud	No Connected Environments or Universal Subscription. Same paid Org/license.		No	On-prem Cribl Stream or Edge → Cribl.Cloud-managed Worker Group for central processing (same license/org).	Standard Edge → Stream or Stream → Stream “hub” architectures.
On-prem → Cribl.Cloud	Different paid Orgs, no Universal Subscription.		Yes	Hybrid Worker Group for Customer A → Managed Worker Group in MSP Org B.	Counts as two separate paid environments.
Hybrid Worker Group → Cribl.Cloud Hybrid or Managed Worker Group	Same Organization		Yes	Cribl.Cloud Hybrid Worker Group in Workspace A → Cribl.Cloud-managed Worker Group in Workspace B.	Typical “hybrid WG in Workspace A → Cloud WG in Workspace B” expansion path.

Sender → Receiver	Org/Workspace scenario	Cribl HTTP/TCP Support	Receiver Billed for Ingest?	Example Deployments	Notes
Hybrid Worker Group → Cribl.Cloud Hybrid or Managed Worker Group	Different Organization		Yes	Hybrid Worker Group for Customer A → Managed Worker Group in MSP Org B.	Counts as two separate paid environments.
On-prem Cribl Edge or Stream → Cribl.Cloud Hybrid Worker Group	Same Org with Universal Subscription		Yes	On-prem Edge Fleet with Universal Subscription → Cribl.Cloud Hybrid Worker Group	Not supported for single-ingest billing. Treated as separate environments.
On-prem Cribl Edge or Stream → Cribl.Cloud-managed Worker Group	Same Org with Universal Subscription		Yes	On-prem Worker Group with Universal Subscription → Cribl.Cloud-managed Worker Group in a Workspace.	US governs billing of on-prem env; Cloud ingest is billed normally unless also covered by shared key/Org rules.
Cross-environment flow on free tier	N/A		Not supported	Free-tier Workspace ↔ paid Org.	Free tier license or subscriptions don't qualify for single-ingest benefit.

Requirements and Caveats

- Both Organizations must have active, paid Cribl.Cloud plans or paid licenses.
- Workers on 4.11.1 and older can't share data with Workers on 4.12.0 and newer.
- For Cribl-managed Workers using **Cribl HTTP** in Cribl.Cloud, ensure **TLS settings match** between the Cribl HTTP Source and Destination (the Source enables TLS by default).

Limitations

You **can't** transfer data across Organizations in these scenarios:

- Workers on 4.11.1 and older communicating with Workers on 4.12.0 and newer.
- Free tier Workers communicating between themselves within the same Workspace.

How We Secure the Data Transfer

A shared-key mechanism validates and establishes trust between Leaders. The shared key will apply when the Leader of the sending environment has the exact same license key installed as the receiving environment, or when any Cribl.Cloud environment has Workspaces set up in the same Organization.

Set up Sources and Destinations to Transfer Data

Prerequisites / Before You Begin

Ensure the following criteria are met before configuring:

- Both of the environments for which you want to transfer data have an active billing contract or paid license.
- You have Leader networking configured to allow cross-environment communication between Edge Nodes.
- You have a Source configured to collect data (events, logs, metrics, files, etc). This Source will send the data to the Cribl TCP or Cribl HTTP Destination on the receiving Organization.
- On Cribl-managed Edge Nodes in Cribl.Cloud, make sure that TLS is either disabled on both the Cribl HTTP Source and the Cribl HTTP Destination it's receiving data from, or enabled on both. Otherwise, no data will flow. In the Source, TLS is enabled by default.

Choose Your Source and Destination Pair

You can send data between Organizations using one of these Source and Destination pairs:

- Cribl TCP Destination and Cribl TCP Source.
- Cribl HTTP Destination and Cribl HTTP Source.

For more information about these two options and how to decide which one is right for your Organization, read Interoperation Protocols.

Configure Cross-Workspace Data Collection



This guide will cover the Cribl TCP setup, but the general workflow is the same for the Cribl HTTP Source

and Destination.

This workflow assumes you already have data ingested in Cribl Edge, and you want to move the data to a Edge Node in another Organization:

1. Configure the Cribl TCP Source in the Organization where you want to **receive** data.
2. Configure the Cribl TCP Destination in the second Organization. This Destination will **send** data to the receiving Organization.
    - In the **Cribl Endpoint** field, enter the Leader hostname in the **Address** field, in one of these formats:
        - Cribl.Cloud: `https://<your-workspace-name>.cribl.cloud`
        - On-prem: `http://localhost:<port>`
3. If your setup meets the criteria (see Key Aspects of Data Transfer for a refresher), you will see data flowing into the Cribl TCP Source from Workers that are not connected to the same Leader. Next!

12.7. Integrate With Other Services

The following topics provide detailed guides and better practice for integrating with different external services.

Source	Description
Cribl Edge as a Sidecar Container in AWS ECS	Cribl Edge as a Sidecar Container in AWS ECS
Amazon Security Lake Integration	Integrate Cribl Edge's Amazon Security Lake Destination with its Namesake Service
Amazon Elastic Kubernetes Services (EKS) Add-On for Edge	Amazon Elastic Kubernetes Services Add-On for Edge

12.7.1. Cribl Edge As A Sidecar Container In AWS ECS

You can use Cribl Edge to collect logs from a service or task running in ECS. This is an alternative to forwarding logs to CloudWatch Logs.

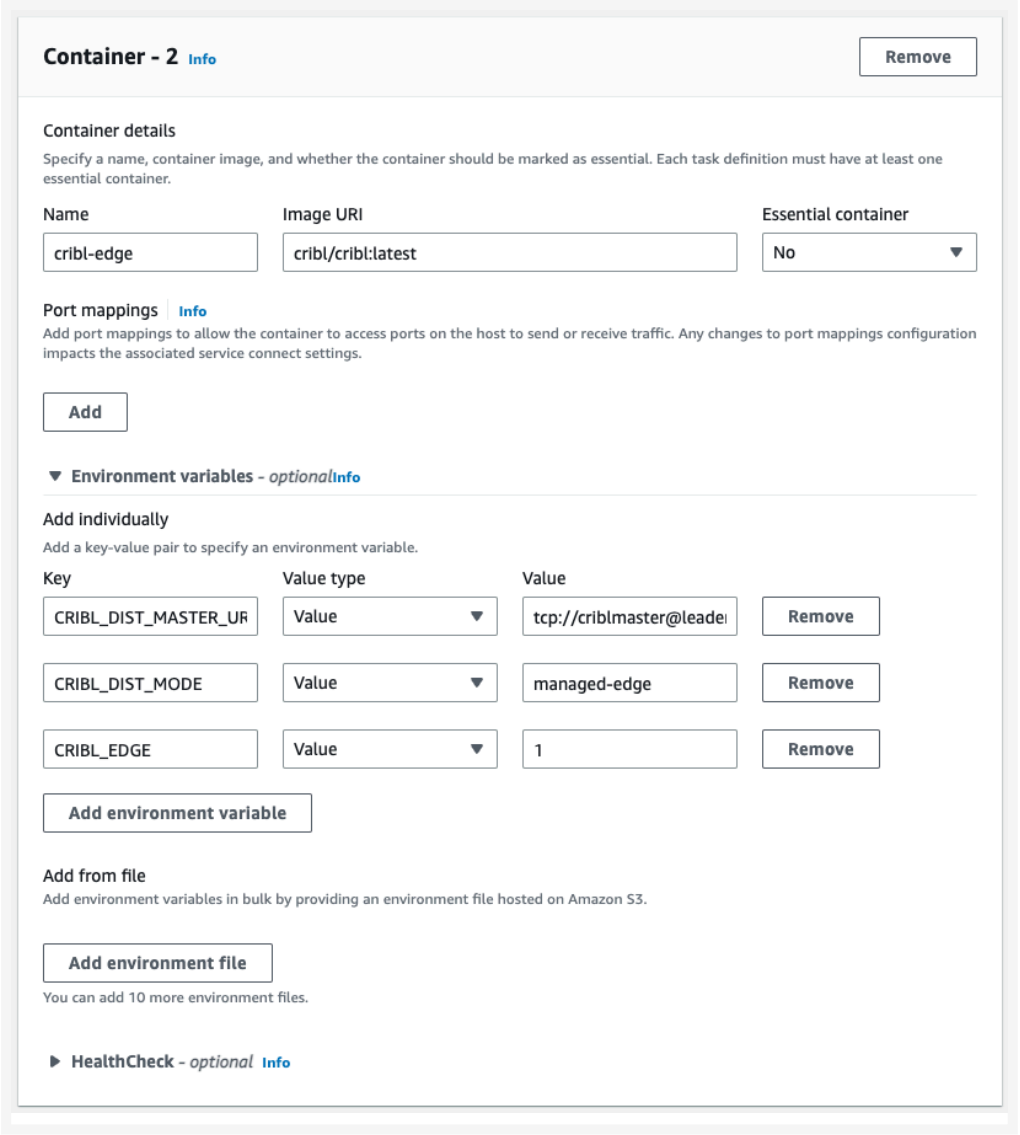


Currently, you can't configure Windows or macOS hosts as sidecar containers in AWS.

Container Definition

Add a secondary container to the task definition. To configure the container, you must first set the following environment variables:

Name	Value
`CRIBL_DIST_LEADER_URL`	`tcp://<token>@<leader>:4200`
`CRIBL_DIST_MODE`	`managed-edge`
`CRIBL_EDGE`	`1`

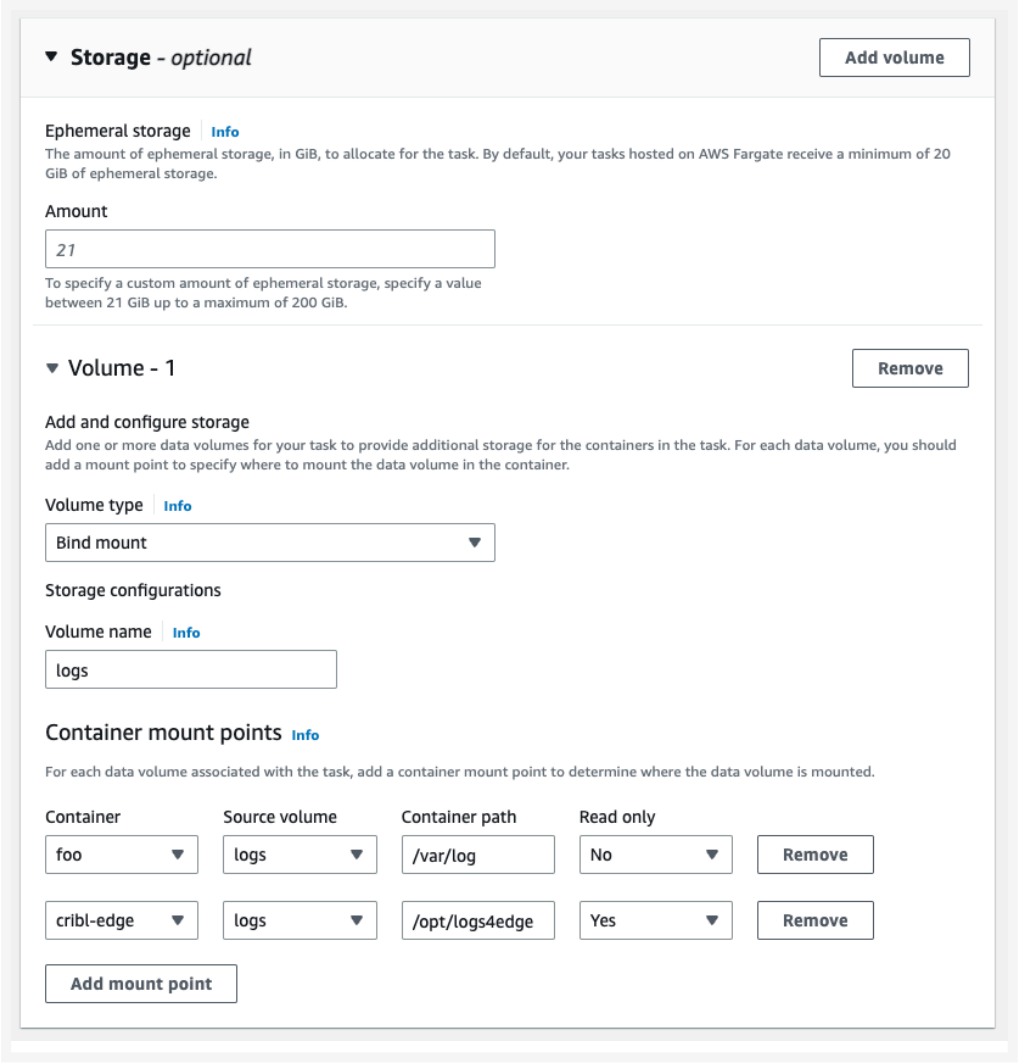


Add Environment Variables to Container Details

Configure Storage

To pass logs between containers, you can use a Bind Mount, which is a special folder mounted between containers. For details, see Using Data Volumes in Tasks.

Here's an example configuration from the ECS task definition:



ECS Task Definition Example

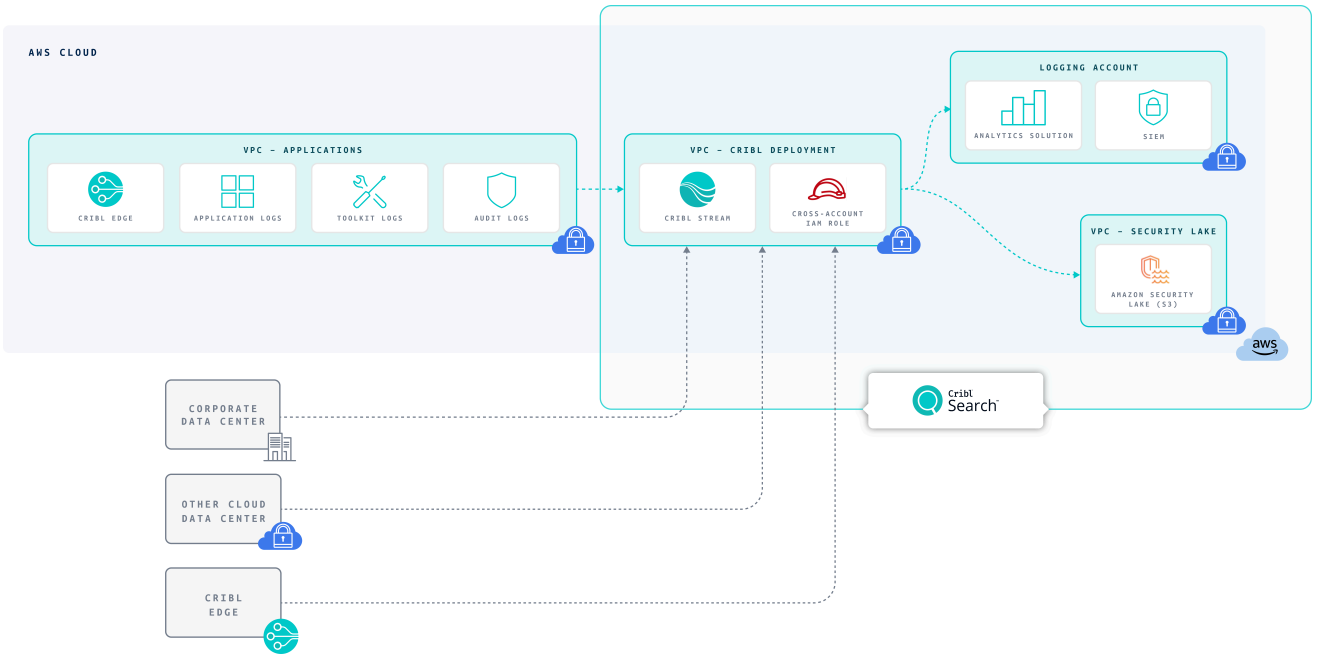
Sample JSON Container Definitions

You can adapt the following sample JSON definitions for the container. Note, this is not a full ECS task definition.

```
{
    "containerDefinitions": [
        {
            "name": "whatever",
            "image": "foo/bar:latest",
            ...
            "mountPoints": [
                {
                    "sourceVolume": "logs4edge",
                    "containerPath": "/var/log",
                    "readOnly": false
                }
            ],
            "volumesFrom": [],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-create-group": "true",
                    "awslogs-group": "/ecs/cribl-leader-plus-edge",
                    "awslogs-region": "us-west-2",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        },
        {
            "name": "cribl-edge",
            "image": "cribl/cribl:latest",
            "cpu": 0,
            "portMappings": [],
            "essential": true,
            "environment": [
                {
                    "name": "CRIBL_EDGE",
                    "value": "1"
                },
                {
                    "name": "CRIBL_DIST_MODE",
                    "value": "managed-edge"
                },
```

12.7.2. Amazon Security Lake Integration

Cribl Edge supports sending data to Amazon Security Lake as Parquet files that adhere to the Open Cybersecurity Schema Framework (OCSF). The diagram below illustrates the all-AWS variant of this architecture.



Integrating Cribl Edge with Amazon Security Lake

This guide walks you through an example where Cribl Edge, running in Cribl.Cloud, sends Palo Alto Networks logs to Amazon Security Lake. The workflow starts with a Cribl Edge Syslog Source, which ingests the Palo Alto data and routes it to a Cribl Edge Amazon Security Lake Destination, which then writes Parquet files to Amazon Security Lake. These Parquet files organize the data according to the OCSF schema for the appropriate OCSF Event Class, in this case Network Activity [4001].

You can work with other data sources besides Palo Alto Networks, and other OCSF Event Classes besides Network Activity [4001]. See Going Beyond This Example below.

If you want to use Amazon Security Lake with a Cribl Edge instance deployed on-prem or in your private cloud, the setup procedures will differ from those documented here. Please contact Cribl via the `#packs` channel of Cribl Community Slack.

Set Up Amazon Security Lake

Complete the Amazon Security Lake Getting Started instructions, paying particular attention to the following actions:

1. Enable Amazon Security Lake for your AWS account.

2. Define the **collection objective** and **target objective**.
3. Select the Regions where you want to create S3 buckets to store the data in OCSF format.

Note your Amazon Security Lake S3 bucket name for use later in the setup process.

Set Up Cribl Edge

Complete the procedures in this section before doing anything else in Cribl Edge.

First, find and note the Amazon Resource Name (ARN) that enables AWS to locate your Cribl Edge instance.

1. Log in to Cribl.Cloud.
2. In the sidebar, select **Workspace**, and then select **Trust**.
3. Select the copy icon next to the **Worker ARN**, and paste the ARN into a local text editor.
4. Copy the 12-digit number in the middle of the ARN. This is the AWS Account ID you'll need later in the setup process.

For example:

- If your ARN is `arn:aws:iam::222233334444:role/main-default`, then …
- Your AWS Account ID will be `222233334444`.

Next, install the Cribl Pack that Cribl Edge will use to send data to Amazon Security Lake in the required OCSF format. (As the reference shows, the Palo Alto Networks data chosen for this example requires the **OCSF Post Processor Pack**.)

1. On the top bar, select **Products**, and then select **Cribl Edge**.
2. Select the name of the Fleet whose Edge Nodes you want to send data to Amazon Security Lake. (For this example, we'll use the `default` Fleet.)
3. Navigate to **Processing** > **Packs** (Stream) or **More** > **Packs** (Edge) and select **Add Pack**. From the drop-down, select **Import from Git**.
4. For the **URL**, enter:

   ```
   https://github.com/asc-me-cribl/cribl_ocsf_postprocessing
   ```

   Copy

5. For the **New Pack ID**, enter:

   ```
   Cribl-OCSF_Post_Processor
   ```

Copy

6. Select **Import** to close the modal. Cribl Edge will take a little time to finish importing the Pack.

At the bottom of the list of Packs, you should now see your newly imported Pack, listed with the display name of **OCSF Post Processor**.

Finally, you will need the Parquet schema that supports the OCSF Event Class you're working with. In this example, that's OCSF Event Class 4001, so as shown in the reference below, you'll need the `OCSF 4001 Network Activity` Parquet schema.

- First, download the Parquet schema from this GitHub repo.
- Then, upload the schema to the Cribl Edge Parquet schema library as described in Parquet Schemas.

Once this is done, the Parquet schema you need should be available when you configure the **Parquet Settings** for your Cribl Edge Amazon Security Lake Destination.

Please post any questions you have about these procedures to the `#packs` channel of Cribl Community Slack.

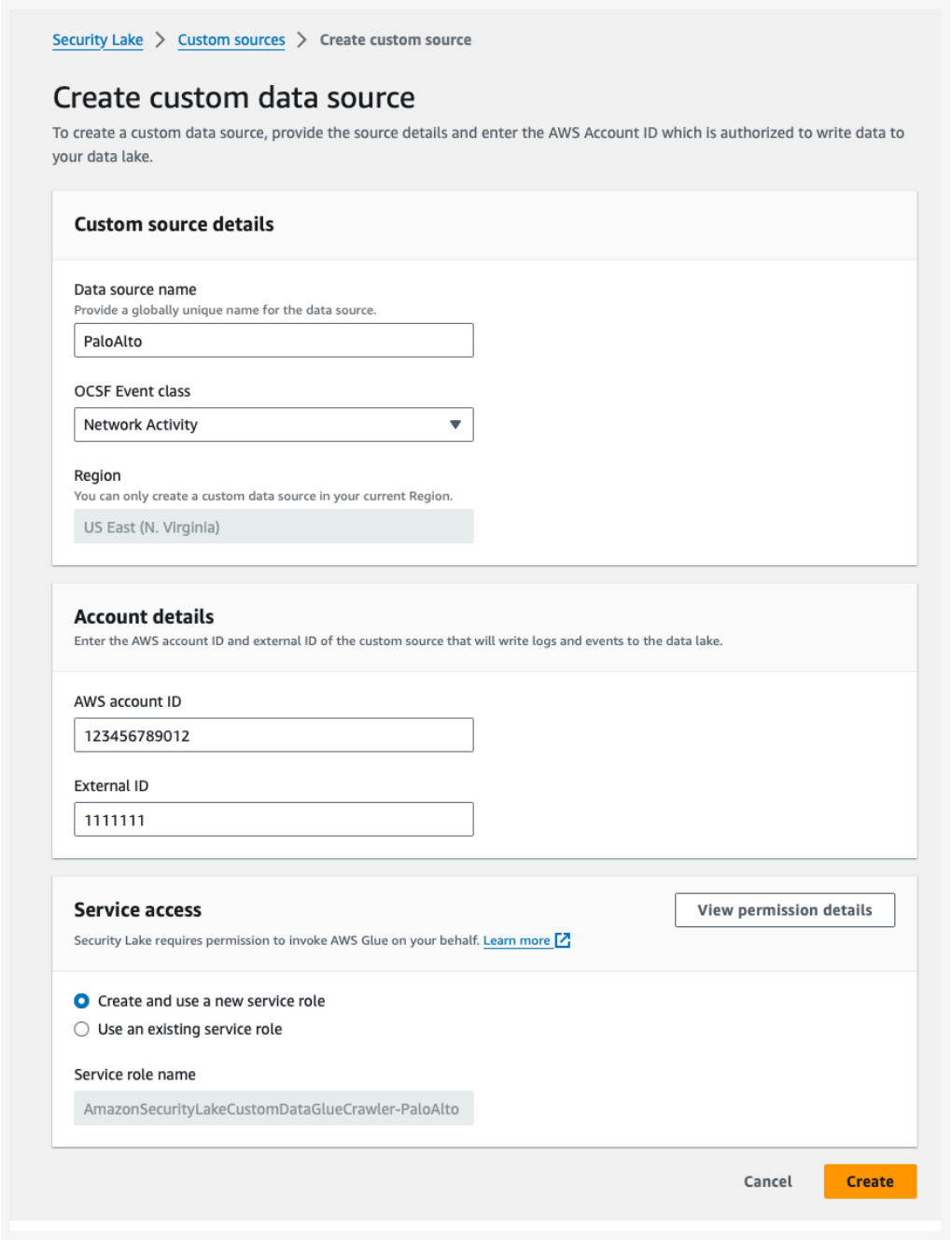
Create an Amazon Security Lake Custom Source

Because Cribl is not a native Amazon service, you'll configure what Amazon calls a **Security Lake custom source**, and the Cribl Edge Amazon Security Lake Destination will write to that.

Before you begin:

1. Have your Cribl.Cloud AWS account ID, which you identified earlier, handy.
2. Ensure that you have permissions in AWS Lake Formation to create AWS Glue databases and tables. For more information, see Granting and revoking permissions using the named resource method in the AWS docs.

Then complete the instructions here in the AWS docs. These docs will take you through the **Create custom data source** template show below.



Creating an Amazon Security Lake custom source

Once the custom source has been created, still in AWS, navigate to **Security Lake** > **Custom sources** and you should see your new custom source in the list.

Now locate the **Provider role ARN**. This is the ARN that Cribl Edge will use to declare its role to AWS.

1. Select the copy icon next to the **Provider role ARN** to add it to your clipboard. Keep the ARN handy for use later in the setup process.
2. Navigate to the **Identity and Access Management (IAM)** screen.
3. From the left menu, select **Access management** > **Roles**.
4. In the **Roles** search box, paste the part of the ARN that begins `AmazonSecurityLake` and ends with the Region.

5. When the role name appears below, select the link.
6. In the **Trust relationships** tab, view the `ExternalId` element in the JSON object that appears there.

This is the External ID for the trust relationship, which you created when went through the template. You will need this ID later in the setup process.

Create a Cribl Amazon Security Lake Destination

Back in Cribl Edge, create a new Amazon Security Lake Destination in the Routing UI as described in the Destination topic. In the **New Destination** modal, configure the settings specified below. For all settings **not** mentioned in the following notes, you can keep the defaults.

General Settings

S3 Bucket name: The S3 bucket name you noted when setting Up Amazon Security Lake.

Region: Select the Region where the S3 bucket is located.

Optional Settings

File name prefix expression: Keep the default (`CriblOut`) if it satisfies your requirements; otherwise, edit to suit your needs.

Authentication

Authentication method must be set to `Auto` (the default).

Assume Role

Cribl strongly recommends using the **Auto** authentication method in conjunction with the **Assume Role** settings as described below. Both Cribl and Amazon discourage the use of other approaches.

When using **Assume Role** to access resources in a different Region than Cribl Edge, you can target the AWS Security Token Service (STS) endpoint specific to that Region by using the `CRIBL_AWS_STS_REGION` environment variable on your Edge Node. To specify a custom STS endpoint, set the `CRIBL_AWS_STS_ENDPOINT` environment variable. If both are set, `CRIBL_AWS_STS_ENDPOINT` takes precedence. Setting an invalid Region or endpoint results in a fallback to the global STS endpoint.

AssumeRole ARN: This is the **Provider role ARN** you copied after creating your custom source in Amazon Security Lake.

External ID: Enter the ID that you specified when creating your Amazon Security Lake custom source.

Processing Settings Post-Processing

Pipeline: From the drop-down, select the Pack you installed earlier:

```
PACK Cribl-OCSF_Post_Processor (OCSF Post Processor)
```

Copy

System fields: Remove any fields specified here.

Parquet Settings

Parquet schema: From the drop-down select `OCSF 4001 Network Activity`, since OCSF Event Class 4001 describes the events coming from Palo Alto Networks in this example.

At this point, you can select **Save**; the defaults for **Advanced Settings** should work fine for this example. Then, **Commit** and **Deploy**.



You must create one Cribl Edge Amazon Security Lake Destination for **each unique pairing** of a data source with an OCSF Event Class.

- In this example, we're ingesting Palo Alto Networks Threat and Traffic data, which requires the OCSF Network Activity [4001] Event Class. That's one pairing.
- If you also wanted to ingest CrowdStrike Network Activity data, that would be a new data source paired with the same OCSF Event Class - such as, **that would be a second unique pairing**. You would need to create a **separate** Cribl Edge Amazon Security Lake Destination for that pairing.

The reference below shows all the possible pairings of data sources and OCSF Event Classes.

Connect a Cribl Source to the S3 Destination

We'll now configure a Cribl Edge Syslog Source to ingest Palo Alto Networks system logs, using the QuickConnect UI. (The reference below shows what Cribl Edge Source to use for each supported data source.)

Navigate to QuickConnect as described here. After selecting the Syslog Source tile, select **Select Existing**, then select the preconfigured `in_syslog` Source. When prompted to `Switch in_syslog to send to QuickConnect instead of Routes?`, select `Yes`.

Select **+** and drag the connection line to the S3 Destination you created above.

The connection between the Syslog Source and the S3 Destination should now be enabled.

Commit and **Deploy** the changes.

Send Data to Amazon Security Lake

Return to your AWS Console. You should see Parquet files landing in the S3 bucket. These files should contain the Palo Alto Networks syslog data you sent through Cribl Edge.

Troubleshooting and Testing

Good things to double-check:

- Are you sending the events to your Amazon Security Lake in Parquet format?
- Are your permissions set properly for your IAM role to write to the S3 bucket?

If the answer to either question is "No," your data will not make it into Amazon Security Lake.

To send sample events from the GitHub repo, using netcat:

1. Download the `oYLbFU.json` sample file.
2. Run the following command, replacing `<cribl_org_id>` with your Cribl.Cloud Organization ID:

```
cat oYLbFU.json | nc default.main.<cribl_org_id>.cribl.cloud 10070
```

Copy

With a Cribl.Cloud Enterprise plan, generalize the above URL's `default.main` substring to `<groupName>.main` when sending to other Fleets.

Beyond This Example

If you want to send data from sources other than Palo Alto Networks, you can adapt the above instructions to use the appropriate Source and Pack (and/or modifications to the OCSF Post Processor Pack) in Cribl Edge. This holds true as long as the OCSF Event Classes you want to work with are among those supported by Cribl Edge. For each unique pairing of a data source with an OCSF Event Class, you'll need to create a separate Amazon Security Lake Destination.

In the next section is a list of the supported data sources, what OCSF Event Classes they handle, and what Packs and Parquet schemas you need to work with them.

Reference: Supported Data Sources

Data Source	OCSF Event Classes Handled	Cribl Source	Cribl Pack Display Name	Parquet Schema(s)

Azure Audit Logs	3002, 3004	REST Collector against Microsoft Graph API	Azure Audit Logs to OCSF	`OCSF 3002 Authentication`
				
				`OCSF 3004 Entity Management`

Azure NSG Flow Logs	4001	Azure Blob Storage Source, run as scheduled jobs, not as a Pull Source	Azure NSG Flow Logs	`OCSF 4001 Network Activity`
Cisco ASA	4001	Syslog	Cisco ASA	`OCSF 4001 Network Activity`
Cisco FTD	4001	Syslog	Cisco FTD	`OCSF 4001 Network Activity`
CrowdStrike Account Change	3001	CrowdStrike FDR	Crowdstrike FDR Pack	`OCSF 3001 Account Change`
CrowdStrike Authentication	3002	CrowdStrike FDR	Crowdstrike FDR Pack	`OCSF 3002 Authentication`
CrowdStrike Network Activity	4001	CrowdStrike FDR	Crowdstrike FDR Pack	`OCSF 4001 Network Activity`
GCP Audit Logs Account Activity	3001	Google Cloud Pub/Sub	Google Cloud Audit Logs	`OCSF 3001 Account Change`
Palo Alto Networks (PAN) Threat and Traffic	4001	Syslog	OCSF Post Processor Pack	`OCSF 4001 Network Activity`

SentinelOne	1001, 2001, 3002, 4001	Amazon S3	SentinelOne Cloud Funnel	`OCSF 1001 File System Activity`
				
				`OCSF 2001 Security Finding`
				
				`OCSF 3002 Authentication`

12.7.3. Amazon Elastic Kubernetes Services (EKS) Add-On For Edge

Use Amazon Elastic Kubernetes Services Add-On to track cost of Kubernetes application in the AWS Cloud.

Amazon Elastic Kubernetes Services Amazon EKS is a managed container service to run and scale Kubernetes applications in the AWS Cloud.

Cribl Edge provides an optimized bundle for Amazon EKS cluster cost visibility. This means you can accurately track costs by namespace, cluster, Pod, or by organizational concepts such as team or application. Kubernetes platform administrators and finance leaders can use Cribl Edge to visualize a breakdown of their Amazon EKS cluster charges. This can help them to allocate costs and chargeback organizational units such as application teams.

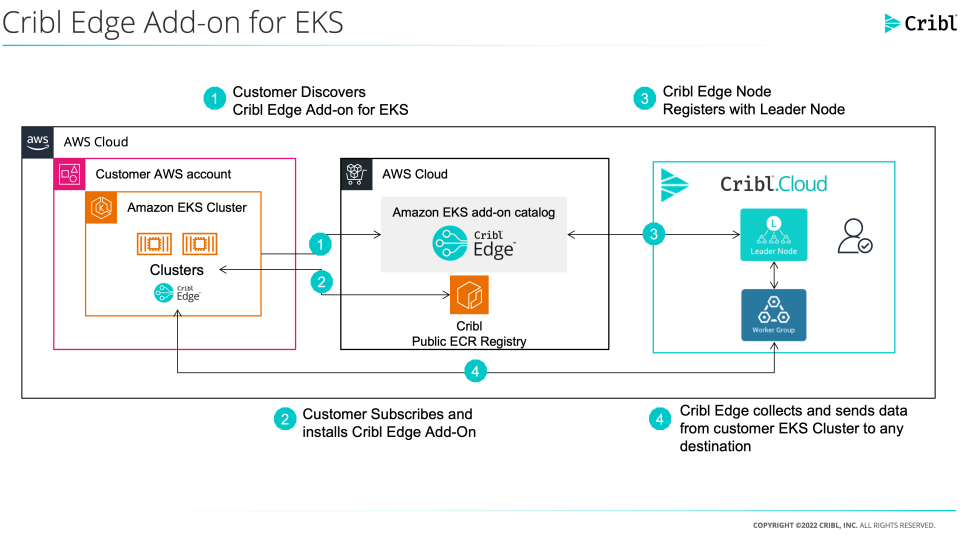
You can use your existing AWS support agreement to obtain support.

In This Article

In this article, you will learn more about how the Amazon EKS architecture interacts with Cribl Edge. You will also learn how to deploy Cribl Edge on EKS using one of two different methods:

1. Deploy Cribl Edge on an Amazon EKS cluster using the Amazon EKS add-on.
2. Deploy Cribl Edge on an Amazon EKS cluster via AWS CLI.

Architecture Overview



Deploy Cribl Edge on an Amazon EKS Cluster

This section covers how to deploy Edge on an AWS EKS cluster.

Prerequisites

To deploy Edge on an Amazon EKS cluster using the Amazon EKS add-on, make sure you have:

- A Cribl.Cloud deployment or a self-hosted Cribl Leader.
- The ability to subscribe to Cribl Edge on AWS Marketplace.
- `kubectl`, AWS CLI, and (optional) `eksctl` installed on your machine.
- Access to an Amazon EKS Cluster.

Set up the Cribl Edge API Server

1. Log into your Cribl Deployment in a self-hosted deployment, or log into Cribl.Cloud, and select **Manage Edge**.
2. In Edge, select the Fleet for which you want to access deployment details.



If you have Fleets running Edge Nodes with version 4.8.2 and older, you can encounter failing health checks that require specific API server configuration. See Common Challenges on the Edge for more information.

Access the Cribl URI and Token

To enable your Cribl Edge add-on for EKS, you'll need the Cribl URI and token. You can collect it from the Kubernetes script in Edge:

1. In Edge, select a Fleet to configure.
2. Select **Add/Update Edge Node** and select **Kubernetes**.
3. In the **Script** field, copy the `cribl.leader` URI, starting at `tls:`. This contains the auth token and the organization name for your deployment.
   Example URI:
   `tls://<auth_token>@<cribl_org_id>.cribl.cloud?group=<default_fleet>`
4. Save this URI somewhere you can easily access it in next steps.

Enable the Cribl Edge Add-On from EKS Console

To get started, access your Amazon EKS console and open your EKS clusters.

1. In the **Add-ons** tab, select **Get more add-ons**.
2. Type `Cribl Edge` into the search bar.
3. Follow the onscreen instructions to enable the Cribl Edge add-on for your Amazon EKS cluster. Learn more about direct deployment to Amazon EKS clusters from this AWS blog post.

Configure AWS EKS Settings

After you enable the Cribl Edge add-on, configure its settings:

1. Under **Version**, select the latest version.
2. Under **Select IAM role**, select **Inherit from node**.
3. Twirl open **Optional configuration settings** and navigate to **Configuration values**.
4. Insert the following JSON configuration:

```
{
 "cribl": {
     "leader": "tls://<token>@<cribl_cloud_leader>.cribl.cloud?group=<default_fl
 }
}
```

Copy

5. Replace the `leader` value with the URI for your Leader Node gathered in Access the Cribl URI and Token.
    - For self-hosted Cribl deployments, enter the publicly available URI and token for your Leader Node.
6. In **Conflict resolution method**, select **None**.

Enable Cribl Edge Add-on Using AWS CLI

You can also enable the Cribl Edge add-on via CLI.

Enter this on the command line:

```
aws eks create-addon --addon-name cribl_cribledge --cluster-name $YOUR_CLUSTER_NAME --
region $AWS_REGION --configuration-values file://path_to_cribl_leader.json
```

Example output:


```
{
    "addon": {
        "addonName": "cribl_cribledge",
        "clusterName": "$YOUR_CLUSTER_NAME",
        "status": "CREATING",
        "addonVersion": " v4.3.1-eksbuild.1",
        "health": {
            "issues": []
        },
        "addonArn": "arn:aws:eks:$AWS_REGION:xxxxxxxxxxxx:addon/$YOUR_CLUSTER_NAME/cr
        "createdAt": "2022-12-01T12:18:26.497000-08:00",
        "modifiedAt": "2022-12-01T12:50:52.222000-08:00",
        "tags": {}
    }
}
```


Copy

To monitor the installation status, you can run the following command:

```
aws eks describe-addon --addon-name cribl_cribledge --cluster-name $YOUR_CLUSTER_NAME
--region $AWS_REGION
```

Example output:

```
{
    "addon": {
        "addonName": "cribl_cribledge",
        "clusterName": "cribl",
        "status": "ACTIVE",
        "addonVersion": "v4.3.1-eksbuild.1",
        "health": {
            "issues": []
        },
        "addonArn": "arn:aws:eks:$AWS_REGION:xxxxxxxxxxxx:addon/cribl/cribl_cribledge
        "createdAt": "2023-11-09T08:28:35.212000-05:00",
        "modifiedAt": "2023-11-09T08:29:44.401000-05:00",
        "tags": {},
        "configurationValues": "{\r\n\t\"cribl\": {\r\n\t     \"leader\": \"tls://cr
    }
}
```


Copy

13. Monitor Health And Metrics

Monitor health and metrics in Cribl Edge.

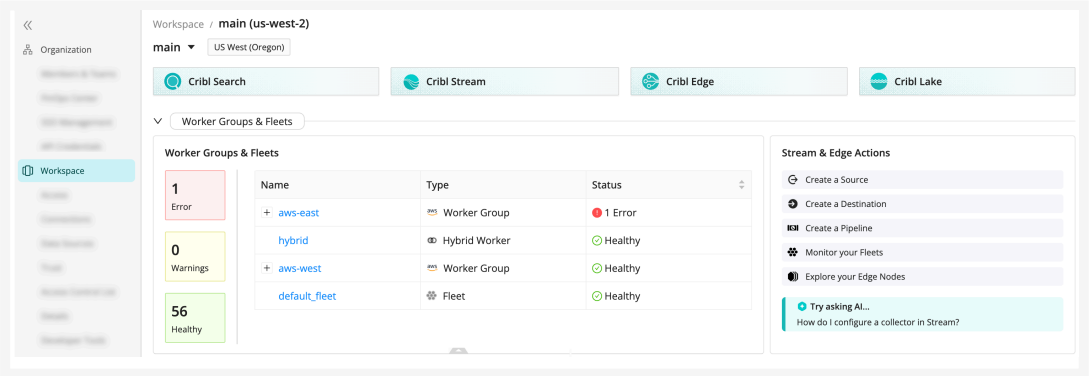
Cribl Edge offers you multiple ways to monitor your deployments in order to get an operational view of the health of your Fleets, Edge Nodes, Sources and Destinations, and more.

Within your Cribl Edge deployment, you can individually monitor the data throughput of different components:

- **Sources**: Data sources that Cribl Edge is receiving data from.
- **Destinations**: The locations Cribl Edge will send processed data.
- **Routes**: The paths that data takes through Cribl Edge.
- **Pipelines**: The collections of Routes that Edge uses to process data.
- **Packs**: Prebuilt Pipelines and configurations you can use to accelerate deployment.
- **Data Fields (Edge Nodes only)**: The specific fields within your data that you can monitor.

Cribl Edge Monitoring Overview

When you first log in to Cribl, you can view the health status of Edge Fleets in the **Worker Groups & Fleets** area.

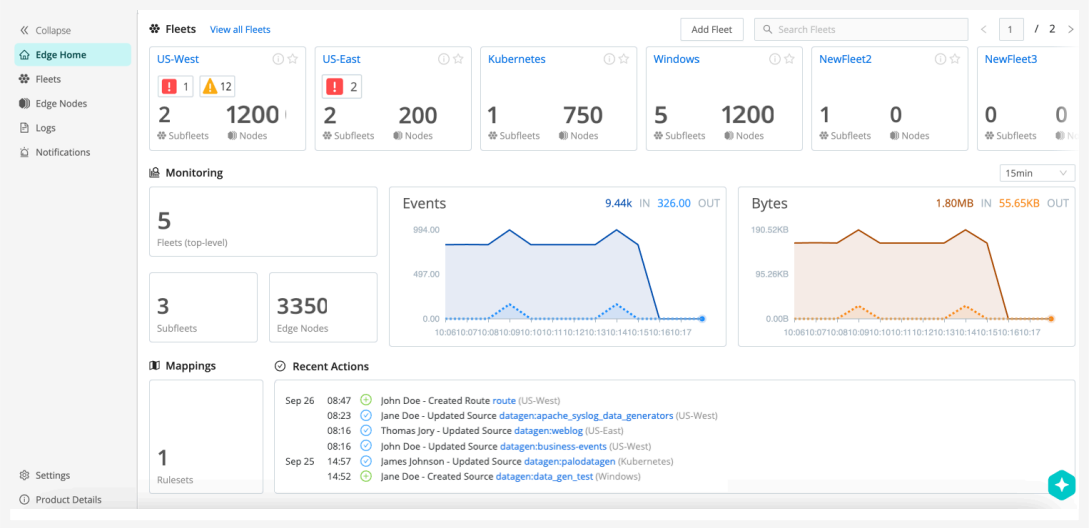


Workspace Landing Page

From this page, select **Cribl Edge** to get to the Edge home page.

Monitor Health on the Edge Home Page

The **Edge Home** page is where you can view aggregate data for all Fleets, Subfleets and Edge Nodes. Through centralized charts, you can monitor real-time traffic flow entering and exiting the system.



Cribl Edge Home page

Analyze Traffic and Timeframes

You can change the time granularity for the **Bytes** chart from the default `15 min`. Select a variety of time ranges from `1 min` up to `1 day` (covering the preceding 24 hours).

The time range settings offer both flexibility and consistency:

- **Configurable time range**: Use the integrated time range picker to analyze activity across different time windows.
- **Smart inheritance**: When opened from a page with a global time picker (for example, **Monitoring** pages or **Fleet** dashboards), the charts inherit that time range by default.
- **Independent selection**: Changes made to the time range within this view are local; they do not affect the global time range of the main page.

For details, see Sources Charts tab and Destination Charts Tab.

Understand the Health Status Icons

You'll see icons next to Sources and Destinations which indicate the health of the resource. Here's what each icon means:

- Green checkmark: Everything is good! All Sources and Destinations are healthy.
- Yellow warning icon: Attention needed. One or more Sources or Destinations are in a warning state.
- Red exclamation point: Critical issue! One or more Sources or Destinations have encountered an error.
- Indeterminate: No data available yet for Sources or Destinations. This usually means you need to enable Basic metrics. See Enable Health Status for instructions.

The Source and Destination health icons are driven by the same internal metrics Edge Nodes send to the Leader. Icons update along with the corresponding metrics. There is no fixed time-based delay that keeps a resource in a warning or error state. Once the underlying errors stop and the next health/metrics update shows recovery, the icon returns to **Healthy**.

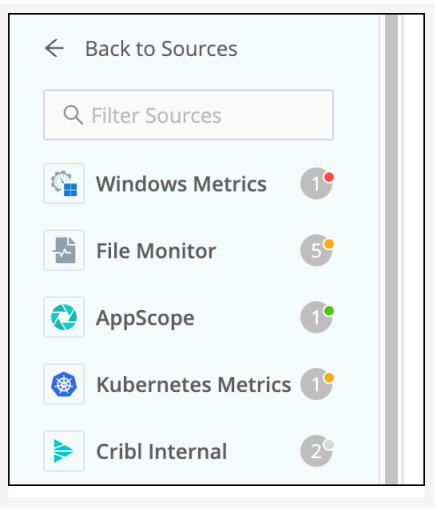
Overall Source and Destination Health

When you add or manage configured Sources and Destinations, you'll see a colored health status (if enabled).

To access the list of configured Sources and Destinations:

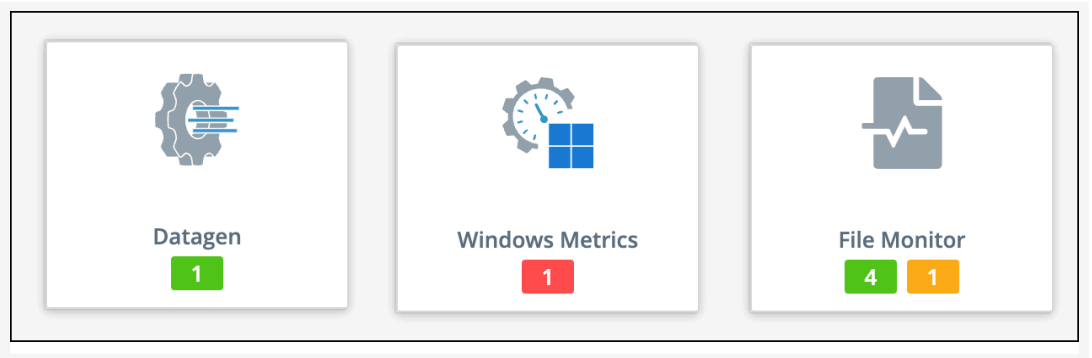
1. Select a **Fleet**.
2. Select **More**, then **Sources** or **Destinations**.

In the list view, the health status is a green, orange, red, or gray dot next to the name, along with a gray circle indicating the number of instances you have configured.



Source list view

In the tile view, colors are applied to the squares that indicate the number of instances you have configured.



Source tile view

To learn more about the health status of any Source or Destination, select its name to open the list of existing Sources or Destinations, and then select the Status icon for the one you want to examine.

Enable Health Status Metrics

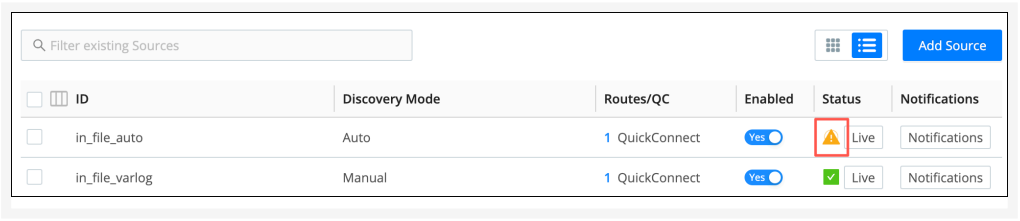
Edge Nodes report the health status of Sources and Destinations to the Leader. To view Source and Destination health status from the **More** drop-down, you must enable health status independently for each Fleet.

You don't have to enable health status to view health icons in the Source and Destination columns (on a Node or Fleet).

To enable health status for a Fleet:

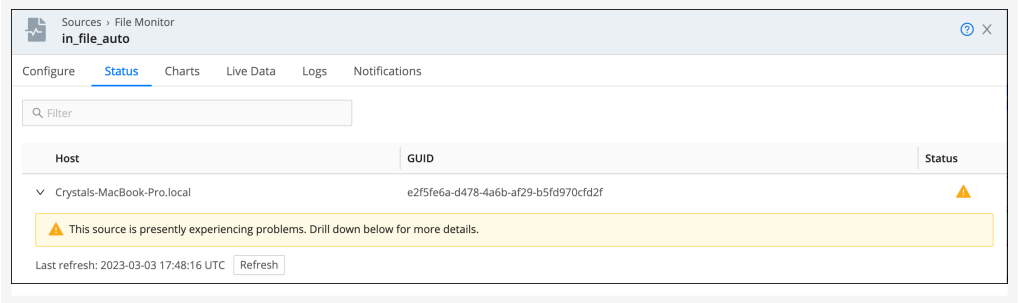
1. Select a Fleet to open it.
2. Select the Fleet Settings tab.
3. Navigate to **General Settings** > **Limits** > **Metrics**.
4. Under **Metrics to send from Edge Nodes**, use the buttons to select or create a set of metrics. You must select at least **Basic** metrics.
5. If you select **Minimal**, Edge Nodes won't report health status for the Fleet. Health icons will appear white for all Sources and Destinations.

To learn more about these metrics options, see Controlling Metrics.



Health status icon

In the modal that opens, select the Status icon again to see more details.



Health status details

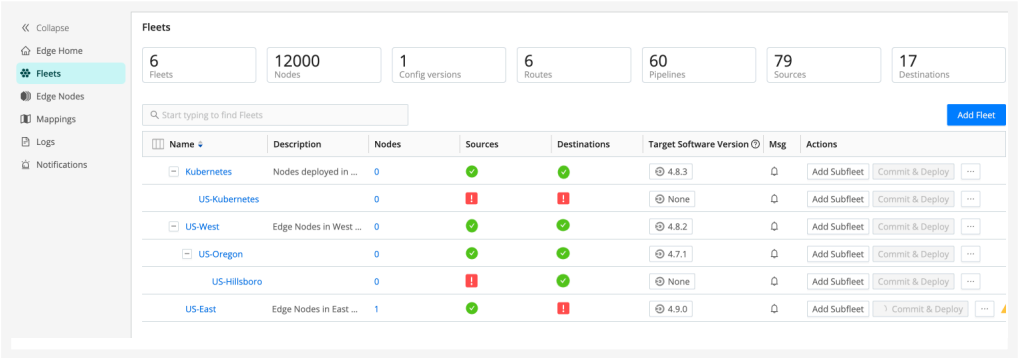
When there are more than 100 Nodes in the Fleet, the **Status** column will not be visible in the list of Nodes. To view the status for an individual Node, expand it in the list. You can only expand one Node at a time. Refresh the list of Nodes and their status with the **Refresh** button.

Charts Tab in Source/Destination Modals

When examining a specific Source or Destination, the Charts tab provides a visualization of recent activity. This tab includes its own time range picker. While it initially inherits the time range from the Monitoring page, you can adjust it independently to zoom in on specific events without changing the settings for your broader monitoring view. For details, see Sources Charts tab and Destination Charts Tab.

Monitor Fleet Information

To view monitoring information for your Fleets, select **Fleets** from the sidebar.



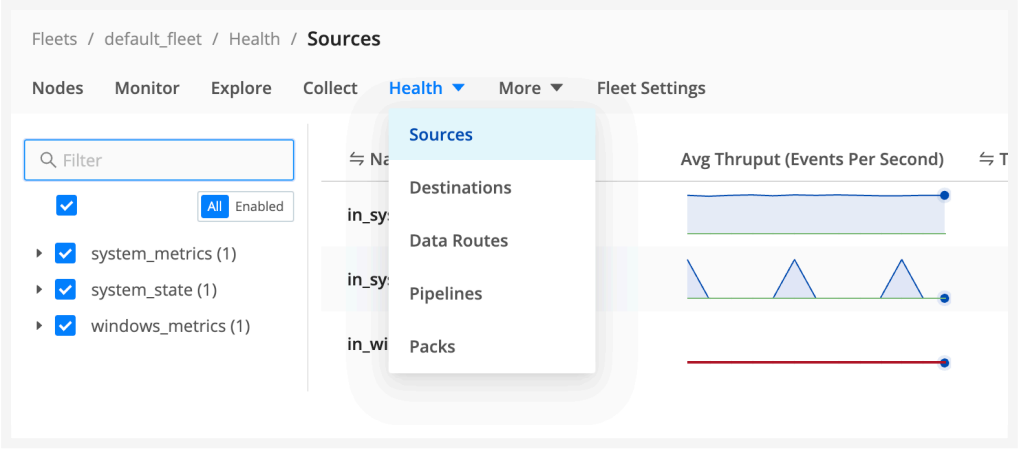
Manage Fleets

The **Fleets** page gives you access to more information about each of your Fleets and the health of Sources and Destinations associated with the Fleets.

You can open an individual **Fleet** using its link, or use the **Search** field.

View Health Statuses for Resources in a Fleet

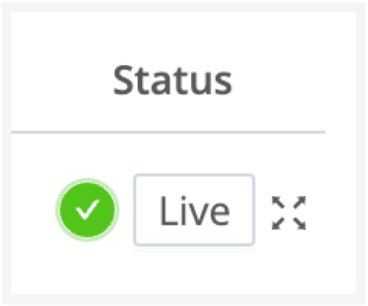
Within any individual Fleet, you can use the **Health** drop-down to view health data for Sources, Destinations, Data Routes, Pipelines, and Packs.



Fleet Health drop-down listing available resource options

In this view, Cribl Edge condenses busy displays to sparklines for legibility.

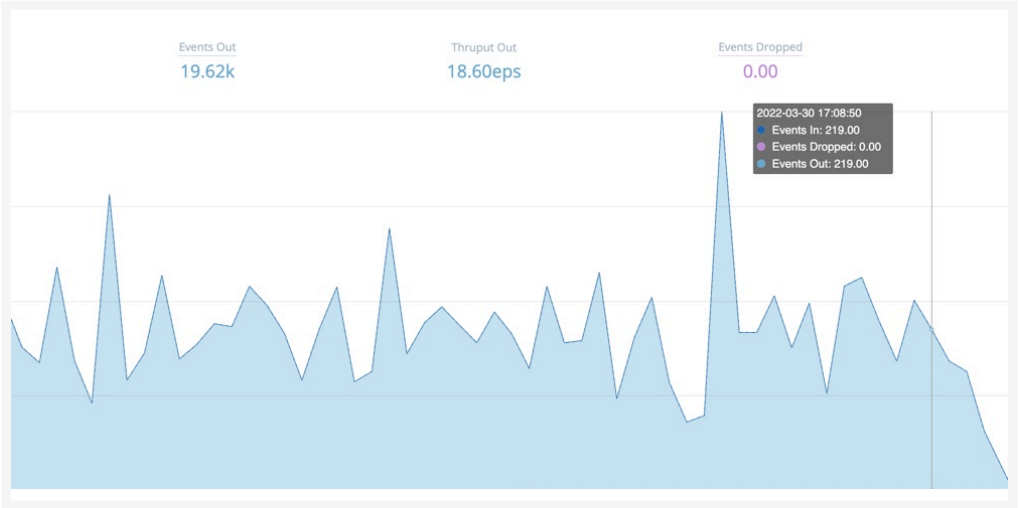
The **Status** column displays the health of each resource with a colored icon. Select the icon to see more details. If a Fleet's **Health** tab is empty, health status might not be enabled because only minimal metrics are being sent. See Enable Health Status for remediation instructions.



Status column with hover activated

To maximize a graph to its detailed version, hover over the right edge to display and select the Maximize icon.

Once you expand a graph, you can hover over it to display further details.



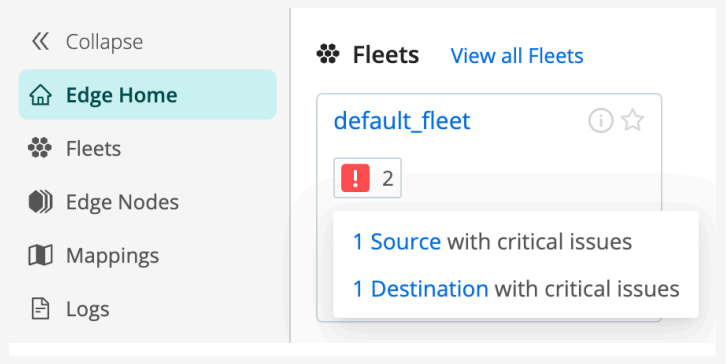
Fleet-level Source and Destination Health Status

To help you identify Fleets that are experiencing issues with Sources or Destinations, Cribl Edge surfaces Fleet-level Source and Destination health status in several places:

- On the **Edge Home** page, on each Fleet tile.
- On the **Fleets** page, in the list of Fleets.
- In each Fleet, in the **Monitor** tab.

To view high-level Source and Destination health status for Fleets:

1. Select **Edge Home** from the sidebar.

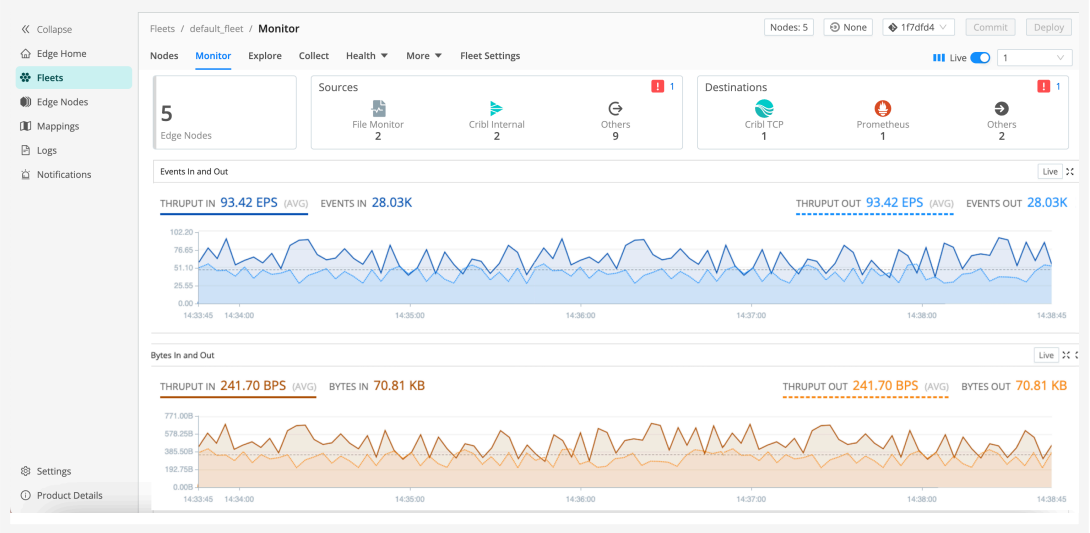


Fleet-level Source and Destination health status icons

2. Select an icon to open the Source or Destination page. The warning and error icons you see here depend on the **Metrics** setting for the Fleet:
    - **Basic metrics**: The Fleet tile will display the number of Sources or Destinations that are experiencing issues. Hover over the icons for a detailed view, and select the Source or Destination.
    - **Minimal metrics**: The Fleet tile will display an icon if there is at least one Source or Destination in a warning state. Selecting an icon will open the Source or Destination page, but you won't see which Sources or Destinations are in a warning or error state.

To view Source and Destination health status on the **Monitoring** page:

1. Select a Fleet, then select Monitor.

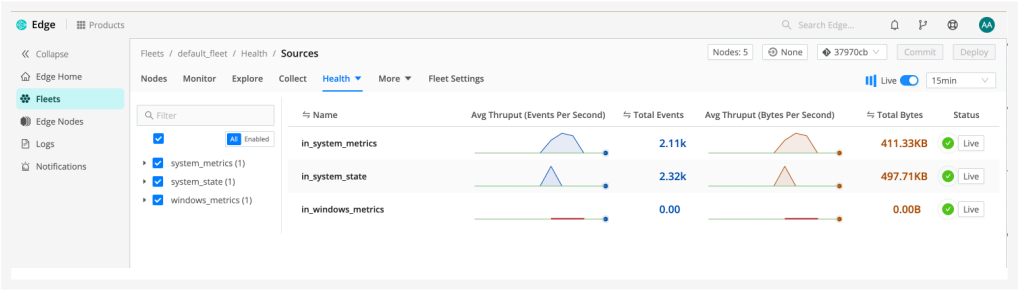


The Monitor tab of a selected Fleet

2. Use this page to identify whether Sources, Destinations, or both are experiencing issues.
3. Select a health status icon to open the Source or Destination list.

Monitor Edge Node Information

To view monitoring information for Edge Nodes, select **Edge Nodes** from the sidebar. Here, you can view the health of each individual Node, as well as Sources and Destinations.



Fleet Health tab

Individual Edge Node Health

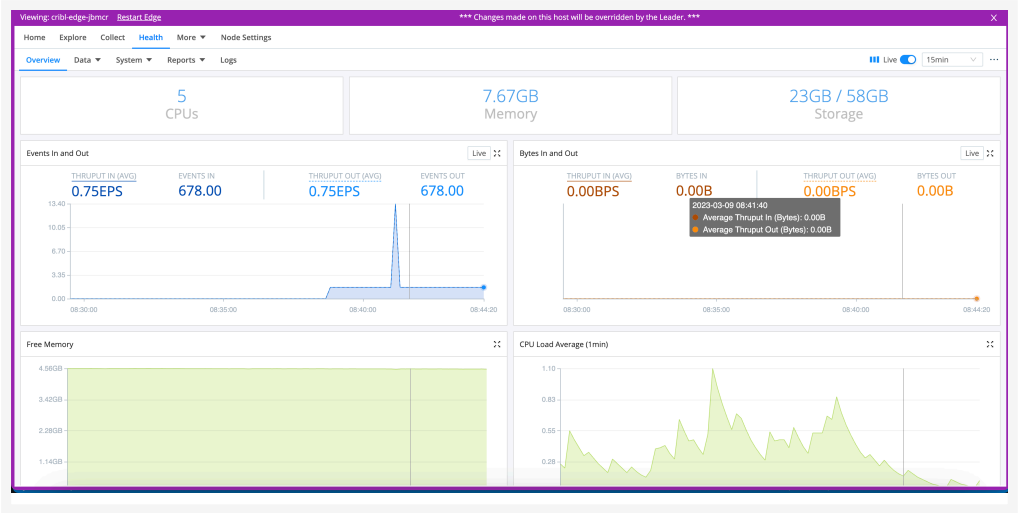
You can review the health status of an Edge Node in two ways: teleport into the Node, or use the Node Info drawer.

Monitor Edge Nodes via Teleport

When you teleport into a Node, you can view monitoring information for **Data Fields**, **System**, **Reports**, and **Logs**.

To teleport into an Edge Node:

1. From the Fleet landing page, select the **List View** tab.
2. Select an Edge Node **GUID** link to teleport from the Leader into the Edge Node.
3. From the top nav of the Edge Node, select the **Health** tab for monitoring details.



Health Overview

The displayed **Storage** represents the amount of free storage remaining on the partition where Cribl Stream is installed. (This quantity might not represent the maximum storage available for the selected Edge Node or Fleet. Also, it does not calculate the system free space.)

Similarly, the **Free Memory** graph reflects only the operating system's `free` statistic, matching Linux's strict `free` definition by excluding `buff/cache` memory. So this graph indicates a lower value than the OS' `available` memory statistic - and it does not necessarily indicate that the OS is running out of memory to allocate.

Byte-related charts show the uncompressed amounts and rates of data processed over the selected time range:

- Events (total) in and out
- Events per second in and out
- Bytes (total) in and out
- Bytes per second in and out

We measure bytes in and out based on the size of `_raw`, if this field is present and is of type `string`. Otherwise, we use the size of read (uncompressed) data.

The displayed **CPU Load Average** is an average Edge Node Process, updated at 1-minute granularity. (It is not an average for the Edge Node as a whole.)

Monitoring data does not persist across Cribl Edge restarts. Keep this in mind before you restart the

server.

Monitor Edge Nodes without Teleporting

To view the Node health status without teleporting, select anywhere on the row to open the Node Info drawer.

System Monitoring

To view System Monitoring information, first teleport into the Node.

From the **Health** tab, open the **System** drop-down to view isolated throughput for:

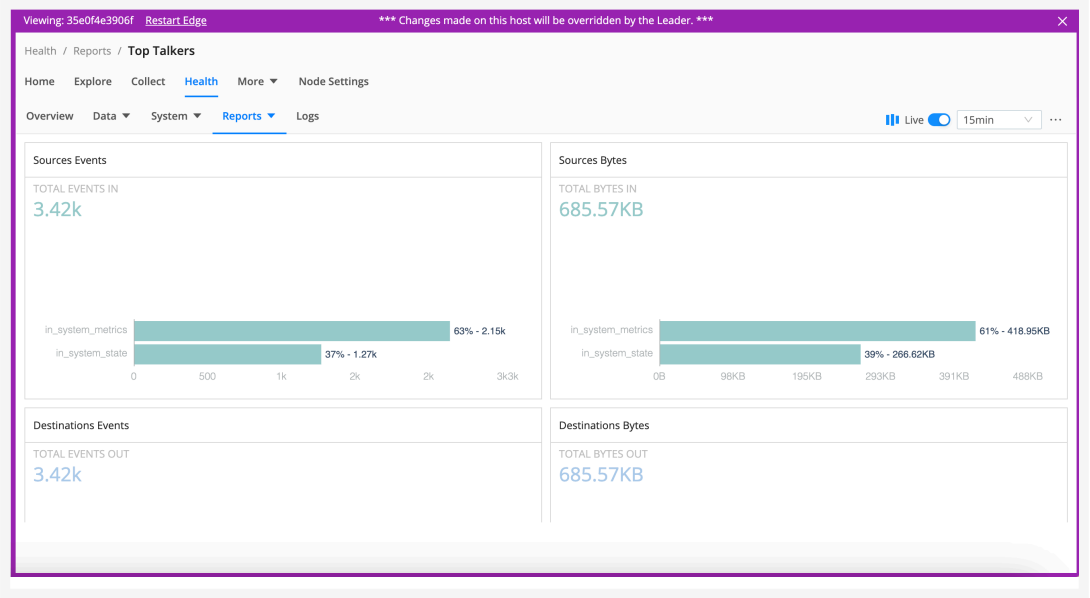
- Queues (Sources)
- Queues (Destinations)

For details, see Persistent Queues.

Reports

To view reports for Top Talkers, first teleport into the Node.

Select **Health** > **Reports** > **Top Talkers**, where you can examine your five highest-volume Sources, Destinations, Pipelines, Routes, and Packs. All components are ranked by events throughput. Sources and Destinations get separate rankings by bytes in and out, respectively.



Top Talkers

Internal Logs and Metrics

To view all logs, select **Logs** from the sidebar. You can also teleport into a Node to view Node-specific logs.

Internal logs and internal metrics provide comprehensive information about an instance's status/health, inputs, outputs, Pipelines, Routes, Functions, and traffic.



Cribl Edge displays metrics in units of KB, MB, GB, and TB. At each level, these are multiples of `1024`.

Health Endpoint

Each Cribl Edge instance exposes a `/health` endpoint that is commonly used along with a load balancer to support operational decision-making. See Leader High Availability/Failover: Load Balancers for more information.



For many HTTP-based Sources, you can enable a Source-level health check endpoint in the **Advanced Settings** tab. The request URL format for these endpoints is `http(s)://${hostName}:${port}/cribl_health`. Refer to the configuration instructions for a Source to learn whether the health check endpoint is available for it.

Types of Logs

Cribl Edge produces several categories of internal logs:

- **API Server Logs**: Generated by the main `API/main` process and written under `$CRIBL_HOME/log/`. For example, `cribl.log`, `access.log`, `audit.log`, `notifications.log`, `ui-access.log`.
- **Edge Node Process Logs**: Generated by Node processes and written under `$CRIBL_HOME/log/node/<N>/…`.
- **Fleet Logs**: Generated by processes that manage configuration for Fleets, typically under `$CRIBL_HOME/log/group/<fleet>/…`.
- **Service Process Logs on Leader Nodes**: Generated by supporting services such as connections, metrics, notifications, and others, typically under `$CRIBL_HOME/log/service/<serviceName>/…`.
- **Stderr-based Logs**: Generated when certain processes encounter fatal errors, including `cribl_stderr.log`.

Use the **Monitoring** > **Logs** page to search across these logs from the UI. For a complete list of internal log types, their locations, and their rotation behavior, see Internal Logs.

Forward Logs and Metrics Externally

Cribl Edge supports forwarding internal logs and metrics to your preferred external monitoring solution. To make internal data available to send out, go to **More** > **Sources** and enable the Cribl Internal Source.

This will send internal logs and metrics down through Routes and Pipelines, just like another data source. Both logs and metrics will have a field called `source`, set to the value `cribl`, which you can use in Routes' filters.

Note that the only logs supported here are Edge Node Process logs (see Types of Logs above). You can, however, use a Script Collector to listen for API Server or Edge Node Fleet events.

For recommendations about useful Cribl metrics to monitor, see Internal Metrics.



CriblMetrics Override

The **Disable field metrics** setting - in **Settings** (top nav) > **System General Settings > Limits** - applies only to metrics sent to the Leader Node. When the **Cribl Internal** Source is enabled, Cribl Edge ignores this **Disable field metrics** setting, and full-fidelity data will flow down the Routes.

Control Metrics Volume

To reduce the volume of metrics sent through Cribl Edge, see options on the Cribl Internal Source.

Disable Field Metrics

To send fewer metrics to the Leader Node, you can disable specific metrics types.

By default, Cribl Edge does not send metrics for the `host`, `source`, `sourcetype`, `index`, and `project` fields. You can remove any of these defaults, or add other fields you do not want to send as metrics.

To do this, specify the fields in the **Disable field metrics** blocklist in **two** places:

- In **Settings** > **Global** > **System** > **General Settings** > **Limits** > **Metrics**.
- Per Fleet, in **Fleet Settings** > **System** > **General Settings** > **Limits** > **Metrics**.

However, when you enable the **Cribl Internal** Source, Cribl Edge ignores this **Disable field metrics** setting, and full-fidelity data will flow down the Routes.

Dropping Metrics

When the number of in-flight requests for sending metrics from Edge Node to Leader exceeds a limit (1,000 requests), Nodes will stop sending metrics. Check `channel="clustercomm"` for a log of dropped metrics if

needed.

You can exclude certain metrics from being dropped due to exceeding limits:

1. Navigate to **Settings** > **System** then **General Settings**.
2. Open **Limits** and select **Metrics**, then enter fields in the **Metrics never-drop list**.

This setting is available only on Edge Nodes.

Metrics Garbage Collection

Metrics garbage collection (GC) runs when the total number of stored metrics exceeds the max metrics limit (default 1,000,000). Metrics are then removed, starting with the oldest ones. This happens both on Workers and Leaders.

You can define how often to run garbage collection on each Fleet. Select **Settings** > **Global** > **System** > **General Settings** > **Limits** > **Metrics** > **Metrics GC period**. The default is 60 seconds.

Metrics Tracking by Worker ID

Typically, Cribl Edge assigns all metrics their own Edge Node ID dimensions so they can be split by Node if needed.

You can define which metrics are **not** assigned an Edge Node ID dimension by adding them to the list at **Settings** > **Global** > **System** > **General Settings** > **Limits** > **Metrics** > **Metrics worker tracking**.

Control Metrics Lag and Disk Usage

If one or more Fleets has a large number of enabled Sources, and clicking into these Sources does not promptly display their status, a workaround is to prevent Cribl Edge from writing metrics to disk. On the Leader (versions 4.1.2 and newer), navigate to **Settings** > **Global Settings** > **System** > **General Settings** > **Limits** > **Storage**, and disable the **Persist metrics** toggle. Then restart the Leader.

If you keep **Persist metrics** enabled, you can use the adjacent **Metrics max disk space** field to control the written metrics' footprint. This threshold defaults to `64 GB`.

Cardinality Limit

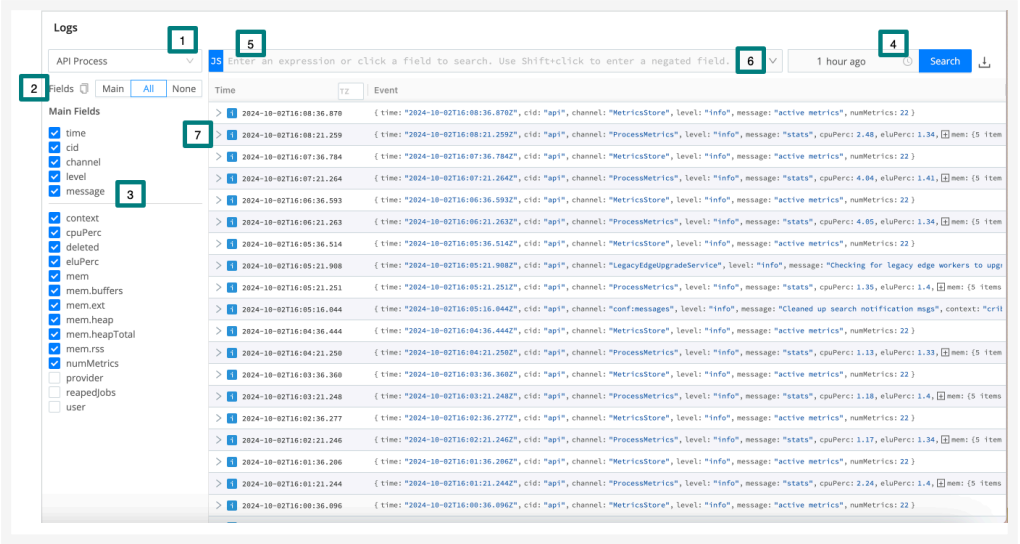
You can define the cardinality limit or metrics on each Fleet. Select **Settings** > **Global Settings** > **System** > **General Settings** > **Limits** > **Metrics** > **Metrics cardinality limit** The default is 1,000.

Refer to **Monitoring** > **Data** > **Data Fields** to identify the fields that have the highest cardinality.

Search Internal Logs

Cribl Edge exists because logs are great and wonderful things! Using Cribl Edge's **Health** > **Logs** page, you can search all Cribl Edge's internal logs at once - from a single location, for both Leader and Edge Node. This enables you to query across all internal logs for strings of interest.

The labels on this screenshot highlight the key controls you can use (see the descriptions below):



Logs page with controls highlighted

Callout	Field	Description
1	Log file selector	Choose the types of logs to view.
2	Fields selector	Select the **Main**, **All**, and **None** toggles to quickly select or deselect multiple check boxes below. Beside these toggles, a **Copy** button enables you to copy field names to the clipboard in CSV format.
3	Fields	Select or deselect these check boxes to determine which columns to display in the **Results** pane at right. (The upper **Main Fields** group will contain data for **every** event; other fields might not display data for all events.)
4	Time range selector	Select a standard or custom range of log data. This picker inherits the global time range by default but operates independently; changing the time here will not update the global page range. (The time range selector uses your local time, even though the logs' timestamps are in UTC.)
5	Search box	To limit the displayed results, enter a JavaScript expression here. An expression must evaluate to `truthy` to return results. You can press **Shift+Enter** to insert a newline. To modify the depth of information that is originally input to the Logs page, see Logging Settings.

Callout	Field	Description
6	Search box history	Select the history arrow to retrieve recent queries.
7	Results	This pane displays most-recent events first. Each event's icon is color-coded to match the event's severity level.

Select individual log events to unwrap an expanded view of their fields.

Logging Settings

On Cribl Edge's Settings pages, you can adjust the level (verbosity) of internal logging data processed, per logging channel. You can also redact fields in customized ways. In a Distributed Deployment, you manage each of these settings on the Edge Node, Fleet Settings, or main Settings for all Fleets.

The logging levels help to categorize the detail and severity of logged information. Choosing the right level ensures you get the information you need for troubleshooting without getting overwhelmed by a flood of data.

Here's a breakdown of each logging level:

Code	Description
`critical`	Logs unexpected failures that need to be resolved immediately.
`error`	Logs only errors which need immediate attention to ensure the system's integrity.
`warn`	Logs warnings about potential issues that might not be critical but warrant attention to prevent future problems.
`info`	Logs general information about the system's operation, providing insights into standard processes and procedures.
`debug`	Logs detailed information for diagnosing and troubleshooting potential problems, providing deep insights into the system's behavior.
`silly`	Logs extremely detailed information and metrics, primarily used for in-depth troubleshooting of inputs and other operations.Use it sparingly and only for short debugging bursts, treating it as "use at your own risk." Enable temporarily, then disable promptly.



Notes About Logs

- By default, all integration logging levels are set to `info`. This means logs of level `info` and above (including `error` and `warn`) will be displayed. Manual adjustments are needed to see `debug` and

`silly` logs.

- Cribl.Cloud users cannot adjust logging levels for Worker Groups or specific integrations.

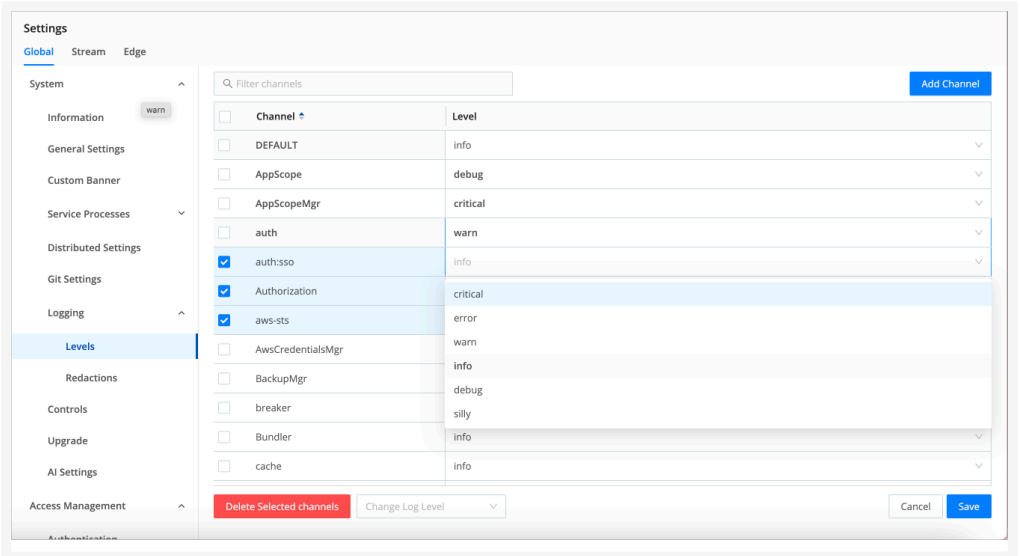
Change Logging Levels

To adjust logging levels:

- From the sidebar, select **Settings** > **Global** > **System** > **Logging** > **Levels**.
- To configure log levels on a Fleet-level, select the Fleet you want to configure. Next, select **Fleet Settings** > **System** > **Logging** > **Levels**.
- In a Single-instance deployment you can adjust settings on a specific Edge Node. First teleport into the Node, then select **Node Settings** > **System** > **Logging** > **Levels**.

On the resulting **Manage Logging Levels** page, you can:

- Modify one channel by selecting its **Level** column. In the resulting drop-down, you can set a verbosity level ranging from **error** up to **debug**. (Top of composite screenshot below.)
- Modify multiple channels by selecting their check boxes, then select the **Change log level** drop-down at the bottom of the page. You can select all channels at once by selecting the top check box. You can search for channels at top right.



Manage Logging Levels page

Change Logging Redactions

On the **Redact Internal Log Fields** page, you can customize the redaction of sensitive, verbose, or just ugly data within Cribl Edge's internal logs. To access these settings:

- In a Single-instance deployment, or for the Leader Node's own logs, select **Settings** > **Global** > **System** > **Logging** > **Redactions**.
- In a Distributed Deployment, select the Fleet you want to configure. Next, select **Fleet Settings** > **System** > **Logging** > **Redactions**.

It's easiest to understand the resulting **Redact Internal Log Fields** page's fields from bottom to top:

Field	Description

13.1. Internal Metrics

When sending Cribl Edge metrics to a metric system of analysis - such as InfluxDB, Splunk, or Elasticsearch - some metrics are particularly valuable. You can use these metrics to set up alerts when Edge Nodes are having problems, a Node is down, a Destination is down, a Source stops providing incoming data, and so on.

Cribl Edge reports its internal metrics within the Cribl Edge UI, in the same way that it reports internal logs. Navigate to **Monitoring** > **Logs** (Cribl Stream), or to a Fleet's **Health** > **Sources** tab (Cribl Edge).

To expose metrics for capture or routing, enable **CriblMetrics** on the Cribl Internal Source.

You can also export metrics (and logs) to JSON files.

By default, Cribl Edge generates internal metrics every 2 seconds. To consume metrics at longer intervals, you can use or adapt the `cribl-metrics_rollup` Pipeline that ships with Cribl Stream. Attach it to your **Cribl Internal** Source as a pre-processing Pipeline. The Pipeline's **Rollup Metrics** Function has a default **Time Window** of 30 seconds, which you can adjust to a different granularity as needed.

You can also use our public endpoints to automate monitoring using your own external tools.

Counter-type metrics in Cribl Edge do not monotonically increase or decrease. They are reset at the end of each reporting period. Cribl Edge does not report counters when their value is 0. For example, if there aren't any Destinations reporting dropped events then the `total.dropped_events` metric won't be reported because its value would be 0.

Breaking Changes

- The `ci` and `co` dimensions are deprecated as of Cribl Edge 4.1.2. See Dimensions for their replacements.
- Disabled Sources no longer generate Cribl internal metrics as of Cribl Edge 4.2.

Total Throughput Metrics

Five important metrics below are prefixed with `total.` These power the top of Cribl Edge's **Monitoring** Dashboard. The first two report on Sources, the remainder on Destinations.

- `total.in_bytes`
- `total.in_events`
- `total.out_events`

- `total.out_bytes`
- `total.dropped_events` - helpful for discovering situations such as: you've disabled a Destination without noticing.

Interpreting Total Metrics

These `total.` metrics' values could reflect Cribl Edge's health, but could also report low activity simply due to the Source system. For example, logs from a store site will be low at low buying periods.

Also, despite the `total.` prefix, these metrics are each specific to the Worker/Edge Node Process that's generating them.

You can distinguish **unique** metrics by their `#input=<id>` dimension. For example, `total.in_events|#input=foo` would be one unique metric, while `total.in_events|#input=bar` would be another.

System Health Metrics

Health metrics are valuable for monitoring system health.

Cribl Edge composite metrics:

- `health.inputs`
- `health.outputs` - see the JSON Examples below for both `health.` metrics.

Hardware or VM infrastructure metrics:

- `system.load_avg`
- `system.free_mem`
- `system.disk_used` - valuable if you know your disk size, especially for monitoring Persistent Queues. Here, a `0` value typically indicates that the disk-usage data provider has not yet provided the metric with data. (Getting the first value should take about one minute.)



The Cribl Internal Source does not export these metrics to Routes or Pipelines, you can obtain them only by using the REST endpoints listed on this page.

All of the above metrics have these possible values:

- `0` = green = healthy.
- `1` = yellow = warning.
- `2` = red = trouble.

Health Inputs/Outputs JSON Examples

The `health.inputs` metrics are reported per Source, and the `health.outputs` metrics per Destination. The `health.inputs` example below has two configured Sources, and two Cribl Stream-internal inputs. The `health.outputs` example includes the built-in `devnull` Destination, and six user-configured Destinations.

Given all the `0` values here, everything is in good shape!

```
"health.inputs": [
  { "model": { "input": "http:http" }}, "val": 0},
  { "model": { "input": "cribl:CriblLogs" }}, "val": 0},
  { "model": { "input": "cribl:CriblMetrics" }}, "val": 0},
  { "model": { "input": "datagen:DatagenWeblog" }}, "val": 0 }
  ],
"health.outputs": [
  { "model": { "output": "devnull:devnull" }}, "val": 0},
  { "model": { "output": "router:MyOut1" }}, "val": 0},
  { "model": { "output": "tcpjson:MyTcpOut1" }}, "val": 0},
  { "model": { "output": "router:MyOut2" }}, "val": 0},
  { "model": { "output": "tcpjson:MyTcpOut2" }}, "val": 0},
  { "model": { "output": "router:MyOut3" }}, "val": 0},
  { "model": { "output": "router:MyOut4" }}, "val": 0 }
  ],
```

Copy

Edge Node Resource Metrics

The Cribl Internal Source reports two useful metrics on the resource usage of individual Edge Node Processes:

- `system.cpu_perc` - CPU percentage usage.
- `system.mem_rss` - RAM usage in bytes.

Edge Node Shutdown Metric

This metric tracks events lost during Worker Process shutdown:

- `shutdown.lost_events` - Number of events lost because they were not flushed before the **Drain timeout** elapsed during Worker Process shutdown, as explained on the Shutdown and Restart Sequence docs.

Persistent Queue Metrics

These metrics are valuable for monitoring persistent queues' behavior:

- `pq.queue_size` - Total bytes in the queue.
- `pq.in_bytes` - Total bytes in the queue for the given **Time Window**.
- `pq.in_events` - Number of events in the queue for the given **Time Window**.
- `pq.out_bytes` - Total bytes flushed from the queue for the given **Time Window**.
- `pq.out_events` - Number of events flushed from the queue for the given **Time Window**.

These are aggregate metrics, but you can distinguish unique metrics per queue. On the Destination side, use the `#output=<id>` dimension. For example, `pq.out_events|#output=kafka` would be one unique metric; `pq.out_events|#output=camus` would be another.

Integration Metrics

Integration metrics provide enhanced observability into the performance and health of Sources and Destinations. See how to control the granularity of integration metrics.

By exposing internal metrics for data flow, the platform helps you understand ingestion rates, queue behavior, connection health, and back-pressure. Depending on your metric granularity level, these I/O metrics provide sufficient signal to troubleshoot slow ingestion, rejected connections, and overloaded Nodes without requiring deep, log-level investigation.



For I/O observability, Cribl Edge uses Detailed metrics by default for all integrations (Sources and Destinations) that support I/O metrics. This standard applies to both Cribl.Cloud and on-prem Organizations.

View Integration Metrics

Metrics are aggregated every minute for timely monitoring. You can monitor integration metrics in three ways:

- **Logs Tab**: In the Sources and Destinations configuration modal, use the dropdown menu to select a specific metrics channel and view the details.
- **Charts Tab**: Performance charts are displayed directly on each supported Source and Destination configuration modal, offering immediate visualization and trend analysis.
- **Through the Cribl Internal Source (CriblMetrics)**: You can query the metrics and build automations using the Internal Cribl Source.

Available Metrics

The following metrics provide a detailed view of Source and Destination performance, covering connection status, request activity, latency, and data throughput:

- `logged.criticals` - Number of criticals logged to the channel.
- `logged.errors` - Number of errors logged to the channel.
- `iometrics.activeCxn` - Number of active inbound connections per Source.
- `iometrics.closeCxn` - Number of closed connections per Source.
- `iometrics.openCxn` - Number of opened connections per Source.
- `iometrics.rejectCxn` - Number of rejected connections per Source. Includes the `reject_reason` dimension (such as `concurrency_limit`, `high_load`, or `ip_validation_failed`).
- `iometrics.consumer_lag` - Offset between the last message in a partition and the consumer's position (Kafka-based Sources only).
- `iometrics.endpoints_healthy_percentage` - Percentage of healthy load-balanced endpoints.
- `iometrics.p95_duration_millis` - 95th percentile of request processing time.
- `iometrics.p99_duration_millis` - 99th percentile of request processing time.
- `iometrics.total_requests` - Number of sent/received requests.
- `iometrics.failed_requests` - Number of failed requests.
- `iometrics.ingest_bps_active` - Histogram of ingestion throughput rate (Bps) for active TCP sockets, S3 file transfers, or HTTP-based sources.
- `iometrics.ingest_bps_closed` – Histogram of lifetime ingestion throughput rate (Bps) for closed TCP sockets and HTTP-based sources.
- `source.in_bytes` - Inbound bytes per Source.
- `source.in_events` - Inbound events per Source.
- `source.out_bytes` - Outbound bytes per Source.
- `source.out_events` - Outbound events per Source.
- `sourcetype.in_bytes` - Inbound bytes per Source type.
- `sourcetype.in_events` - Inbound events per Source type.
- `sourcetype.out_bytes` - Outbound bytes per Source type.
- `sourcetype.out_events` - Outbound events per Source type.
- `total.activeCxn` - Total number of active inbound connections from all HTTP and TCP-based Sources.
- `total.closeCxn` - Total number of closed connections from all HTTP and TCP-based Sources.

- `total.openCxn` - Total number of opened connections from all HTTP and TCP-based Sources.
- `total.rejectCxn` - Total number of rejected connections from all HTTP and TCP-based Sources.

Other Internal Metrics

The Cribl Internal Source emits other metrics that, when displayed in downstream dashboards, can be useful for understanding Cribl Edge's behavior and health. These include:

- `pipe.in_events` - Inbound events per Pipeline.
- `pipe.out_events` - Outbound events per Pipeline.
- `pipe.dropped_events` - Dropped events per Pipeline.
- `metrics_pool.num_metrics` - The total number of unique metrics that have been allocated into memory.
- `collector_cache.size` - Each Collector function (`default/cribl/collectors/<collector>/index.js`) is loaded/initialized only once per job, and then cached. This metric represents the current size of this cache.
- `cluster.metrics.sender.inflight` - Number of metric packets currently being sent from a Worker/Edge Node Process to the API Process, via IPC (interprocess communication).
- `backpressure.outputs` - Destinations experiencing backpressure, causing events to be either blocked or dropped.
- `blocked.outputs` - Blocked Destinations. (This metric is more restrictive than the one listed just above.)
- `pq.queue_size` - Current queue size, in bytes, per Worker Process.
- `host.in_bytes` - Inbound bytes from a given host (host is a characteristic of the data).
- `host.in_events` - Inbound events from a given host (host is a characteristic of the data).
- `host.out_bytes` - Outbound bytes from a given host (host is a characteristic of the data).
- `host.out_events` - Outbound events from a given host (host is a characteristic of the data).
- `index.in_bytes` - Inbound bytes per index.
- `index.in_events` - Inbound events per index.
- `index.out_bytes` - Outbound bytes per index.
- `index.out_events` - Outbound events per index.
- `route.in_bytes` - Inbound bytes per Route.
- `route.in_events` - Inbound events per Route.
- `route.out_bytes` - Outbound bytes per Route.
- `route.out_events` - Outbound events per Route.

- `nodecluster.primary.messages` - Number of IPC (interprocess communication) calls made by a Worker/Edge Node Process to the API Process.
- `cribl.logstream.project.in_events` - Inbound events per Project.
- `cribl.logstream.project.out_events`- Outbound events per Project.
- `cribl.logstream.project.dropped_events` - Dropped events per Project.
- `cribl.logstream.project.in_bytes` - Inbound bytes per Project.
- `cribl.logstream.project.out_byte` - Outbound bytes per Project.
- `cribl.logstream.subscription.in_bytes` - Inbound bytes per Subscription.
- `cribl.logstream.subscription.out_bytes` - Outbound bytes per Subscription.
- `cribl.logstream.subscription.in_events` - Inbound events per Subscription.
- `cribl.logstream.subscription.out_events` - Outbound events per Subscription.

Dimensions

Cribl Edge internal metrics have extra dimensions. These include:

- `cribl_wp` - Identifies the Worker/Edge Node Process that processed the event.
- `event_host` - Machine's hostname from which the event was made.
- `event_source` - Set as `cribl`.
- `failure_reason` - Whether the failed request is `retryable` or `non_retryable`.
- `group` - Name of Cribl Fleet from which the event was made.
- `input` - Entire **Input ID**, including the prefix, from which the event was made.
- `output` - Entire **Output ID**, including the prefix, from which the event was made.
- `status_code` - For failed requests, the status code for the error (`4xx` or `5xx`).
- `reject_reason` - Reason for a rejected connection.
- `ci` - Deprecated; use `input` instead. Cribl plans to drop support for this dimension after version 4.x.
- `co` - Deprecated; use `output` instead. Cribl plans to drop support for this dimension after version 4.x.

Control Metrics

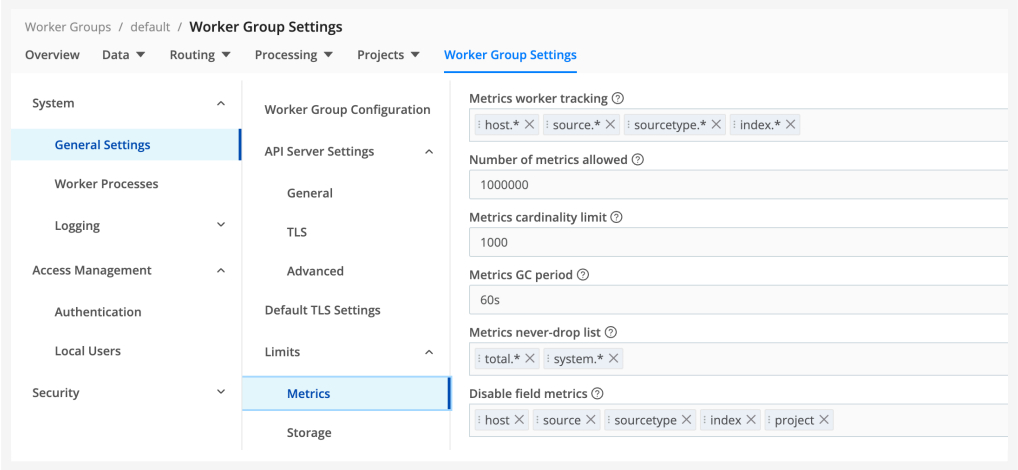
You can control the volume of internal metrics that Cribl Edge emits, by adjusting metrics limits. You access these as follows, per deployment type:

- Single-instance: **Settings** > **System** > **General Settings** > **Limits** > **Metrics**.
- Distributed Leader: **Settings** > **Global Settings** > **System** > **General Settings** > **Limits** > **Metrics**.

- Distributed Fleets: [Select a Fleet] > **Worker Group/Fleet Settings** > **System** > **General Settings** > **Limits** > **Metrics**.

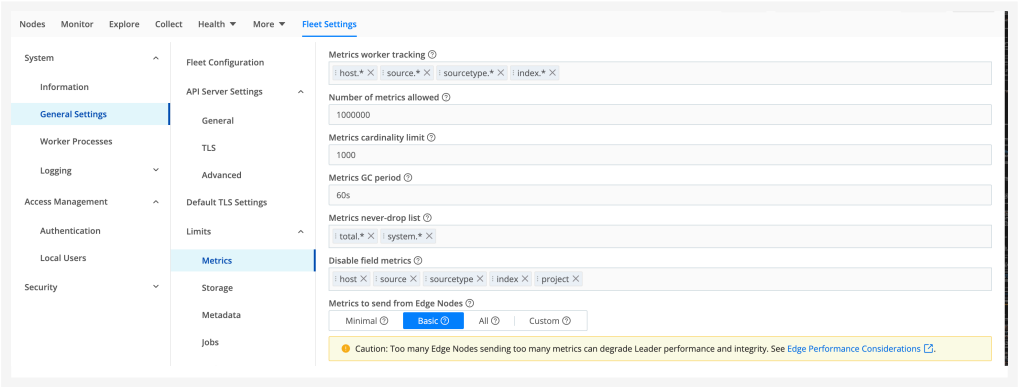
Here, you can adjust the maximum number of metric series allowed, metrics' cardinality, metrics to always include (**Metrics never-drop list**), metrics to always exclude in order to optimize performance (**Disable field metrics**), and other options.

Here's how these controls look in Cribl Stream:



Metrics settings for Cribl Stream Worker Groups

They're somewhat different in Cribl Edge - see the next section for an explanation:



Metrics settings for Cribl Edge Fleets

Control Integration Metrics in Cribl Stream

On Cribl.Cloud instances, you can control the granularity of integration metrics emitted by the Internal Metrics Source to optimize observability while managing system performance and resource usage.

Access the **Metrics Levels** setting by navigating to the **Worker Group** > **Worker Group/Fleet Settings** > **System** > **Metrics Levels**.

Only `Admin` and `Editor` roles can view or modify these settings.

On the **Metrics Limits** page, you'll find a table listing all configured integrations. For each Source or Destination, you can select the desired metrics level from a dropdown.

Level	Purpose	List of Metrics Sent

Minimal	Reports total throughput metrics, offering a high-level view of overall data flow without integration-specific details. Ideal for low-cardinality environments.	`logged.errors`, `logged.criticals`
		
		`total.openCxn`, `total.closeCxn`, `total.activeCxn`, `total.in_events`, `total.out_events`, `total.in_bytes`, `total.out_bytes`, `total.events`, and `total.bytes`
		
		`iometrics.endpoints_healthy_percentage`, `iometrics.openCxn`, `iometrics.closeCxn`, `iometrics.activeCxn`, and `iometrics.rejectCxn`.

Basic (default)	Includes health metrics, emphasizing error tracking, endpoint health, and success rates. Designed for monitoring the operational state of integrations.	`health.inputs` `iometrics.total_requests`, and `iometrics.failed_requests`.
Detailed	Includes latency metrics, enabling precise performance analysis by capturing percentile-based processing times. **Warning**: This level increases system load.	`iometrics.p95_duration_millis` and `iometrics.p99_duration_millis`.

Control Metrics in Cribl Edge

Cribl Edge lets you choose what metrics each Fleet sends to its Leader Node. Your goal should be to collect the metrics most important to you, within the limits of the Leader's capacity to process them. This is a matter of balancing Fleet cardinality and Leader performance, as explained in the Deployment Planning topic.

Use the **Metrics to send from Edge Nodes** buttons to select one of the metrics sets listed in the table below.

Set	Purpose	List of Metrics Sent
Minimal	Sends metrics for events and bytes in and out. Recommended for high-cardinality deployments.	Sent as aggregate totals per Fleet: `total.in_events`, `total.out_events`, `total.in_bytes`, `total.out_bytes` excluding metrics that are coming directly from Sources or Destinations.

Basic (default)	Contains all the metrics in the minimal set, and more. Sends metrics required to display all monitoring data available in the UI.	Sent as aggregate totals per Source/Destination:
		
		`total.in_events`, `total.out_events`, `total.in_bytes`, `total.out_bytes`.
		
		Also, Sent as aggregate totals per Source:
		
		`health.inputs`
		
		Sent as aggregate totals per Destination:
		
		`health.outputs`
		
		Sent as aggregate totals per Route:
		
		`route.in_events`, `route.in_bytes`, `route.out_events`, `route.out_bytes`, `route.dropped_events`
		
		Sent as aggregate totals per Pipeline:
		
		`pipe.in_events`, `pipe.out_events`, `pipe.dropped_events`
		
		Sent as aggregate totals per Pack:
		
		`pack.in_events`, `pack.out_events`, `pack.err_events`, `pack.dropped_events`

All	Send all metrics with no filtering.	All metrics without filtering, including **Basic**, plus Worker/Edge Node Resource Metrics, Persistent Queue Metrics, and Other Internal Metrics.
Custom	Send a customized set of metrics.	All metrics that match a JavaScript expression that you define, plus the metrics in the **Minimal** set.

Basic is the default selection, and includes the **Minimal** set of metrics. Every Edge Fleet always sends at least **Minimal** metrics, and selecting one of the other options adds more metrics.

Choosing the **Minimal** metrics set has the following effects in the Edge UI:

- Only one UI page, **Edge Home** > **Monitor**, will be fully populated. For the metrics that the **Minimal** set excludes, other pages will show zero values. This applies to the **[Select a Fleet]** > **Health** pages for

Sources, **Destinations**, **Routes**, **Pipelines**, and **Packs**.

- In the **[Select a Fleet]** > **More** pages for **Sources** and **Destinations**, the health indicators in the tile and list views do not function when you select the **Minimal** set. Rather than red, green, or gray, they'll always show white, because the relevant metrics, `health.inputs` and `health.outputs`, will not be sent to the Leader.

The simplest form of a **Custom** set just specifies additional desired metrics by name, using an expression of the form: `name === "metric.name"`. Among the most useful metrics to consider adding are `system.load_avg`, `system.free_mem`, `system.disk_used`, `system.cpu_perc`, and `system.mem_rss`.



Custom Metrics and Their Impact on Monitoring

Enabling custom metrics switches monitoring from **Basic** to **Minimal**. This hides monitoring data for Fleets and Edge Nodes, giving the impression that data flow has stopped. To ensure system monitoring features work as expected with custom metrics, include the **Basic** metrics set within the custom configuration.

Other Metrics Endpoints

Below is basic information on using the `/system/metrics` endpoint, the `/system/info` endpoint, and the `cribl_wp` dimension.

`/system/metrics` Endpoint

`/system/metrics` is Cribl Edge's primary public metrics endpoint, which returns most internal metrics. Many of these retrieved metrics report configuration only, not runtime behavior. For details, see our API Docs.

`/system/info` Endpoint

`/system/info` generates the JSON displayed in the Cribl Edge UI at **Settings** > **Global Settings** > **Diagnostics** > **System Info**. Its two most useful properties are `loadavg` and `memory`.

`loadavg` Example

```
"loadavg": [1.39599609375, 1.22265625, 1.31494140625],
```

This property is an array containing the 1-, 5-, and 15-minute load averages at the UNIX OS level. (On Windows, the return value is always `[0, 0, 0]`.) For details, see the Node.js os.loadavg() documentation.

`memory` Example

```
"memory": { "free": 936206336, "total": 16672968704 },
```

Divide `total` / `free` to monitor memory pressure. If the result exceeds 90%, this indicates a risky situation: you're running out of memory.

`cpus` Alternative

13.2. Internal Logs

Cribl Edge generates internal application logs that monitor its own operations and health. They provide valuable insights into the system's behavior, performance, and potential issues.

Distributed deployments emit a larger set of logs than single-instance deployments. We'll describe the distributed set first.

You can display and export all internal logs by selecting **Monitoring** in the sidebar, then selecting **Logs** (Stream) or selecting **Logs** in the sidebar (Edge). Logs' persistence depends on event volume, not time - for details, see Log Rotation and Retention.

> Several logs listed on this page are exposed only in customer-managed (on-prem) deployments. In Cribl.Cloud, Leaders support Cribl Edge Worker Node logs on hybrid Workers.
>
> However, Organization Members who have the Admin Permission on Cribl Search can use that product to search the `cribl_internal_logs` Dataset for additional details about the Leader and its Cribl-managed Workers.

Leader Node Logs (Distributed)

The `API/main` process emits the following logs into the Leader Node's `$CRIBL_HOME/log/` directory.

Logfile Name	Description	Equivalent on Logs page
`cribl.log`	Principal log in Cribl Edge. Includes telemetry/license-validation logs. Corresponds to top-level `cribl.log` on Diag page.	**Leader** > **API Process**
`access.log`	API calls, for example, `GET /api/v1/version/info`.	**Leader** > **Access**
`audit.log`	Actions pertaining to files, for example, `create`, `update`, `commit`, `deploy`, `delete`.	**Leader** > **Audit**
`notifications.log`	Messages that appear in the Notification list in the UI. For each event, the `time` and `_time` fields record the start of the monitored interval (matching `starttime`), not when the Notification was written.	**Leader** > **Notifications**
`ui-access.log`	Interactions with different UI components described as URLs, for example, `/settings/apidocs`, `/dashboard/logs`.	**Leader** > **UI Access**

The `API/main` process emits the following service logs into the Leader Node's `$CRIBL_HOME/log/service/` directory. Each service includes a `cribl.log` file that logs the service's internal telemetry and an `access.log` file that logs which API calls the service has handled.

Service Name	Description	Equivalent on Logs page
Connections Service	Handles all Worker connections and communication, including heartbeats, bundle deploys, teleporting, restarting, and so on. Workers are assigned to connection processes using a round-robin algorithm.	**Leader** > **Connections Service**
Lease Renewal Service	Handles lease renewal for the primary Leader Node.	**Leader** > **Lease Renewal Service**
Metrics Service	Handles in-memory metrics, merging of incoming packets, metrics persistence and rehydration, and UI queries for metrics.	**Leader** > **Metrics Service**
Notifications Service	Triggers Notifications based on its configuration.	**Leader** > **Notifications Service**

The Config Helper process for each Worker Group/Fleet emits the following log in `$CRIBL_HOME/log/group/GROUPNAME`.

Logfile Name	Description	Equivalent on Logs page
`cribl.log`	Messages about config maintenance, previews, and so on.	**GROUPNAME** > **Config helper**

Outpost Logs

The Outpost service emits the following logs into the `$CRIBL_HOME/log/service/outpost/N/` directory, where `N` is the Outpost Node Process ID:

Logfile Name	Description	Equivalent on the Logs tab in Outpost info drawer
`cribl.log`	Internal telemetry logs.	**Outpost Service N**
`metrics.log`	Outpost Node metrics.	**Outpost Service N - Metrics**

Edge Node Logs (Distributed)

The API Process emits the following log in `$CRIBL_HOME/log/`.

Logfile Name	Description	Equivalent on Logs page
`cribl.log`	Messages about the Worker/Edge Node communicating with the Leader Node (that is, with its API Process) and other API requests, for example, sending metrics, reaping job artifacts.	**GROUPNAME** > **Worker:HOSTNAME** > **API Process**

Each Worker Process emits the following logs in `$CRIBL_HOME/log/worker/N/`, where `N` is the Worker/Edge Node Process ID. (The `metrics.log` file is written only when HTTP-based Destinations exist.)

Logfile Name	Description	Equivalent on Logs page
`cribl.log`	Messages about the Worker/Edge Node processing data.	**GROUPNAME** > **Worker:HOSTNAME** > **Worker Process N**
`metrics.log`	Messages about the Worker/Edge Node's outbound HTTP request statistics.	**GROUPNAME** > **Worker:HOSTNAME** > **Worker Process N**

If TCP load balancing is enabled in at least one supported Source (such as Syslog, TCP JSON, or Cribl TCP), the Worker Process also emits the following log to `$CRIBL_HOME/log/worker/LB`:

Logfile Name	Description	Equivalent on Logs page
`cribl.log`	Internal telemetry of the load balancing process.	**GROUPNAME** > **Worker:HOSTNAME** > **Load Balancer**

Load balancing logs use different channel names to indicate the Source they come from than Worker Process logs. To search by channel name, use `channel == "<source-id>_lb"` instead of `channel == "input:<source-id>"`. For example, for the built-in TCP JSON Source, enter `channel == "in_tcp_json_lb"` in the search box instead of `channel == "input:in_tcp_json"`.

For convenience, the UI aggregates the Worker/Edge Node Process logs as follows.

Logfile Name	Description	Equivalent on Logs page
N/A	Aggregation of all the `Worker Process N` logs and the API Process log.	**GROUPNAME** > **WORKER_NAME**

In Cribl Edge, the logs listed above are currently available only on customer-managed hybrid Workers.

> The single-instance logs listed below are not relevant to Cribl.Cloud.

Single-Instance Logs

The `API/main` process emits the same logs as it does for a Distributed deployment, in `$CRIBL_HOME/log/`:

- `cribl.log`
- `access.log`
- `audit.log`
- `notifications.log`
- `ui-access.log`

Each Worker/Edge Node Process emits the following logs in `$CRIBL_HOME/log/worker/N/`, where `N` is the Worker/Edge Node Process ID. (The `metrics.log` file is written only when HTTP-based Destinations exist.)

Logfile Name	Description	Equivalent on Logs page
`cribl.log`	Messages about the Worker/Edge Node processing data.	**GROUPNAME** > **Worker:HOSTNAME** > **Worker Process N**
`metrics.log`	Messages about the Worker/Edge Node's outbound HTTP request statistics.	**GROUPNAME** > **Worker:HOSTNAME** > **Worker Process N**

`_raw stats` Event Fields

Each Worker/Edge Node Process logs this information at a 1-minute frequency. So each event's scope covers only that Worker/Edge Node Process, over a 1-minute time span defined by the `startTime` and `endTime` fields.

Sample Event


```
{"time":"2022-11-17T16:54:05.349Z","cid":"w0","channel":"server","level":"info","mess
```


Copy

Field Descriptions

Field	Description
`abortCxn`	Number of TCP connections that were aborted.
`activeCxn`	Number of TCP connections newly opened at the time the `_raw` stats are logged. (This is a gauge when exported in internal metrics, and can otherwise be ignored as an instantaneous measurement. Only some application protocols count toward this; for example, any HTTP-based Source does not count.)
`activeEP`	Number of currently active event processors (EPs). EPs are used to process events through Breakers and Pipelines as the events are received from Sources and sent to destinations. EPs are typically created per TCP connection (such as for HTTP).
`blockedEP`	Number of currently blocked event processors (caused by blocking Destinations).
`closeCxn`	Number of TCP connections that were closed.
`cpuPerc`	CPU utilization from the combined user and system activity over the last 60 seconds.
`droppedEvents`	This is equivalent to the `total.dropped_events` metric. Drops can occur from Pipeline Functions, from Destination Backpressure, or from other errors. Any event not sent to a Destination is considered dropped.
`eluPerc`	Event loop utilization. Represents the percentage of time over the last 60 seconds that the NodeJS runtime spent processing events within its event loop.
`endTime`	The end of the timespan represented by these metrics. (Will always be 60 seconds after `startTime`.)
`inBytes`	Number of bytes received from all Sources (based only off `_raw`).
`inEvents`	Number of events received from all inputs after Event Breakers are applied. This can be larger than `outEvents` if events are dropped via Drop, Aggregation, Suppression, Sampling, or similar Functions.
`mem.buffers`	Memory allocated for ArrayBuffers and SharedArrayBuffers.
`mem.ext`	External section of process memory, in MB.
`mem.heap`	Used heap section of process memory, in MB.
`mem.heapTotal`	Total heap section of process memory, in MB.
`mem.rss`	Resident set size section of process memory, in MB.
`openCxn`	Same as `activeCxn`, but tracked as a counter rather than a gauge. So `openCxn` will show all connections newly opened each minute, and is more accurate than using `activeCxn`.

Field	Description
`outBytes`	Number of bytes sent to all Destinations (based only off `_raw`).
`outEvents`	Number of events sent out to all Destinations. This can be larger than `inEvents` due to creating event clones or entirely new unique events (such as when using the Aggregation Function).
`pqInBytes`	Number of bytes that were written to persistent queues, across all Destinations.
`pqInEvents`	Number of events that were written to persistent queues, across all Destinations.
`pqOutBytes`	Number of bytes that were flushed from persistent queues, across all Destinations.
`pqOutEvents`	Number of events that were flushed from persistent queues, across all Destinations.
`pqTotalBytes`	Amount of data currently stored in persistent queues, across all Destinations.
`rejectCxn`	Number of TCP connections that were rejected.
`startTime`	The beginning of the timespan represented by these metrics.
`tasksCompleted`	The number of tasks the process has started and completed for all collection jobs for which it was executing tasks.
`tasksStarted`	The number of tasks the process started for all collection jobs for which it was executing tasks.

Log Rotation and Retention

Cribl Edge writes multiple internal log files for Leaders, Edge Nodes, and supporting services. To prevent any single log file from consuming excessive disk space, Cribl Edge automatically rotates these files based on size.

By default, Cribl Edge:

- Rotates each log file when it reaches 5 MB.
- Keeps the five most recent rotated files per log.

This rotation policy applies to all internal log files, including:

- Leader/Node logs in `$CRIBL_HOME/log/`, such as:
    - `cribl.log`
    - `access.log`
    - `audit.log`
    - `notifications.log`

    - `ui-access.log`
- Service logs in `$CRIBL_HOME/log/service/…`.
- Fleet logs in `$CRIBL_HOME/log/group/<GROUPNAME>/cribl.log`.
- Edge Node logs in `$CRIBL_HOME/log/worker/<N>/…`, such as `metrics.log`.
- Stderr-based logs such as `cribl_stderr.log`.

Verbose logging levels (for example, `debug` or `silly`) and high system activity can cause logs to reach the 5 MB threshold more quickly.



For guidance on forwarding internal logs and metrics to external observability platforms, see Forward Logs and Metrics Externally.

`cribl_stderr` and per-process stderr logs

In rare cases, Cribl Edge's Node.js backend can encounter a fatal error (such as an out-of-memory condition) that prevents normal logging. To help with troubleshooting, Cribl Edge writes details about these failures to special stderr log files, including:

- `cribl_stderr.log` at the node level.
- Per-process stderr log files (for example, `cribl_stderr_<PID>.log`) written for specific processes such as API or service processes.

Key points:

- These stderr logs are intended for Cribl Support and are not exposed in the UI.
- They follow the same size-based rotation policy as other internal logs (rotate at 5 MB, keep five rotated copies per file).
- Some stderr logs include prepended UTC timestamps to make it easier to correlate failures with other system activity.

13.3. Track Data Activity For An Edge Node

Monitor and troubleshoot Edge Node activity.

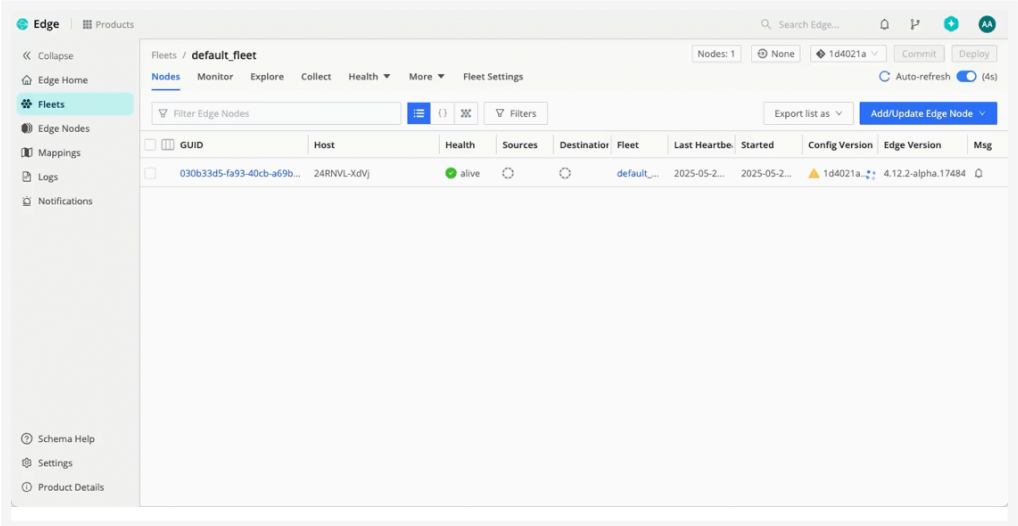
Monitoring and troubleshooting your Edge Nodes guarantees the integrity and reliability of the data you're collecting from your Nodes.

Monitoring helps you detect failures early and take quick corrective actions. Troubleshooting ensures that any problems are not only fixed but also analyzed for root causes to prevent further occurrences.

Cribl Edge provides you with Edge Node data activity, so you can monitor and troubleshoot any issues. You can use this data activity from the Edge Node to find out if any of the associated Sources, Destinations, Routes, Pipelines, or Packs are having issues. Then, drill down into individual elements to identify the potential issues. You can see all of this information without teleporting into the Edge Node.

Access Additional Node Information in the Node Info Panel

The Edge Node information and activity drawer gives you more information about a specific Node. To display additional details and controls, click each row. The **Node Info** panel will open.



Node info panel

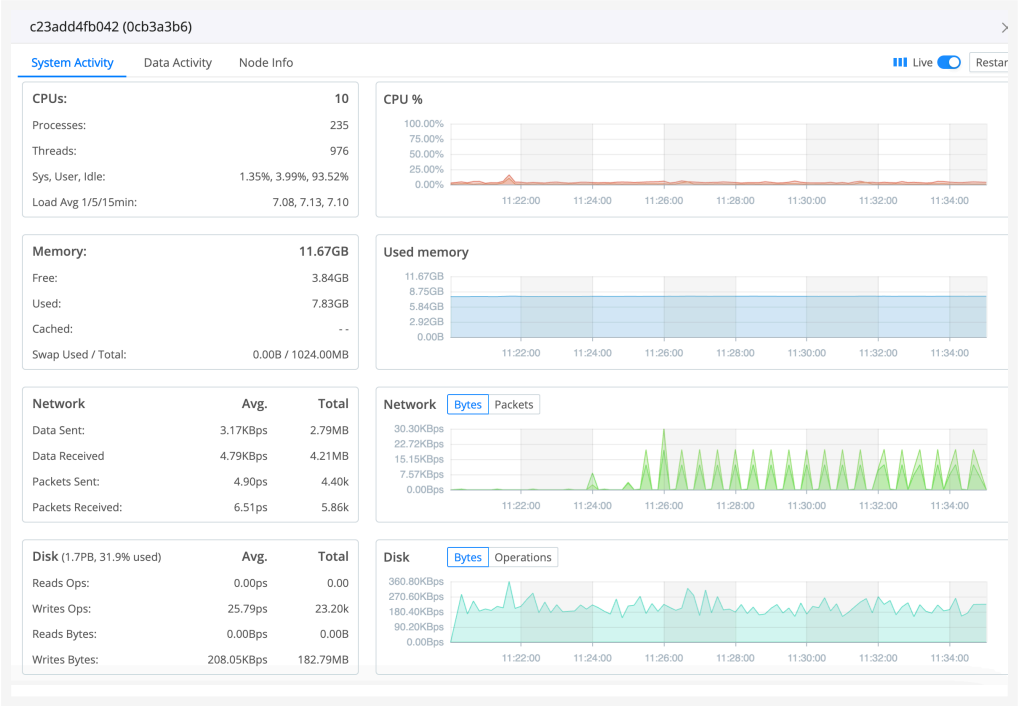
With **Enable teleporting to Nodes** enabled in Fleet Settings, the GUID becomes a link. To open the node information panel in this case, click anywhere in the row except the GUID link.

View the System Activity for a Node

You can monitor system metrics (like CPU, memory, disk usage, and more) for a specific Edge Node using the **System Activity** tab.

Before you can view system metrics for a Node, you must enable the System Metrics Source or Windows Metrics Source for your Fleet.

Once system data is flowing from one of the Metrics Sources, select the row for an Edge Node to view its metrics.



Node info panel system activity

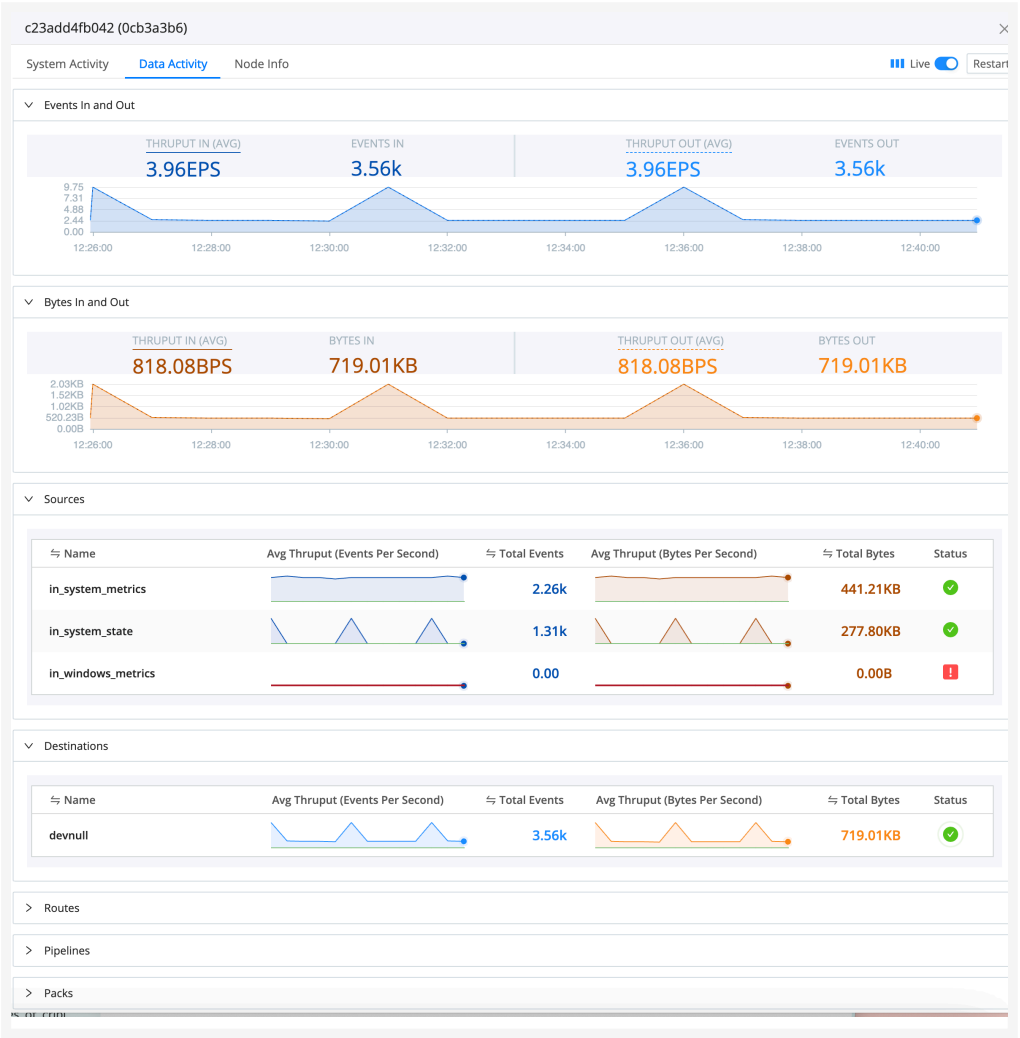
The **System Activity** tab displays the following system information:

Field	Description
CPUs	Total number of CPUs for this Node, processes, threads, percentages of Sys, User, and Idle, and load average over 1, 5, and 15 minutes.
CPU percentage	A graph displaying the total available CPU and CPU usage, in two-minute increments.
Memory	Total memory used by this Node, as well as free, used, cached, and swap used/total memory.
Used memory	A graph displaying the used memory over two-minute increments.
Network	Average and total network data sent and received, and packets sent and received.

Field	Description
Network bytes and packets	A graph displaying network bytes and packets sent and received by the Node, in two-minute increments.
Disk	Average and total read and write operations and read and write bytes.
Disk bytes and operations	A graph displaying disk bytes and operations in two-minute increments.

View the Data Activity for a Node

To view data activity, click the row for an Edge Node.



Node info panel data activity

The **Data Activity** tab displays the following data information:

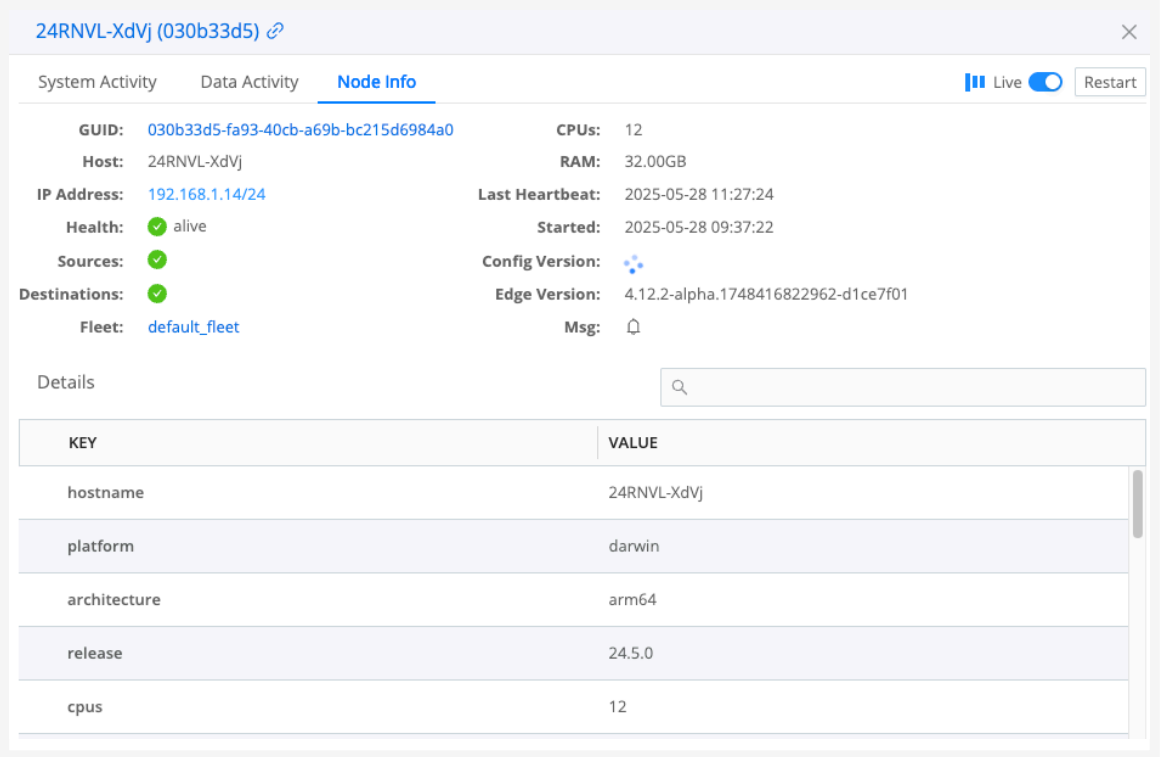
Field	Description
Events In and Out	A graph displaying event throughput in (avg), events in, throughput out (avg) and events out.
Bytes In and Out	A graph displaying bytes throughput in (avg), bytes in, throughput out (avg) and bytes out.
Sources	Each Source that this Node is receiving events from, along with: • The Source name, average throughput (events per second), total events, average byte throughput (bytes per second), and total bytes. • The Source health status, which is useful for identifying Nodes experiencing Source issues and for diagnosing issues with Sources on specific Edge Nodes.
Destination	Each Destination that this Node is sending events to, along with: • The Destination name, average throughput (events per second), total events, average byte throughput (bytes per second), and total bytes. • The Destination health status, which is useful for identifying Nodes experiencing Destination issues and for diagnosing issues with Destinations on specific Edge Nodes.
Routes	The Routes configured on this Node, along with average throughput (events per second and bytes per second), total events, and total bytes.
Pipelines	The Pipelines this Node sends events through, along with average throughput (events per second in and out) and total events in and out.
Packs	The Packs associated with this Node, along with average throughput (events per second in and out) and total events in and out.

The health status icons indicate the overall health of the Sources, Destinations, Routes, Pipelines, and Packs:

- Green checkmark: Everything is good! All resources are healthy.
- Yellow warning icon: Attention needed. One or more resources are in a warning state.
- Red exclamation point: Critical issue! One or more resources have encountered an error.
- Indeterminate: No data available yet.

View Additional Edge Node Info

To view detailed information, click the row for an Edge Node.



Node info panel data activity

The **Node Info** tab displays the following Node information:

Field	Description
Node info	Detailed information for this Node: • GUID • CPUs • Host • RAM • IP Address • Last Heartbeat • Health • Started • Sources • Destinations • Fleet • Config version • Edge version • Msg

14. Work With Data

Cribl Edge is an agent for collecting logs, metrics, and application data and delivering them to Cribl Stream or any supported Destination.

Cribl Edge also offers data processing and routing capabilities that let you transform, enrich, reduce, and redact received data before sending it to one or more target Destinations.

Routes let you control the flow of events by filtering and directing them to selected Pipelines.

Pipelines are where the data transformation happens. Each Pipeline, a series of Functions, can modify the incoming events in a highly configurable way.

Cribl Edge offers tools to help you test and verify your data flow. Use Datagens to generate sample data, and check Data Preview to see and validate the transformation you are performing.

Packs offer portable sets of pre-packaged data routing and transformation tools can include ready-made Routes, Pipelines, Function, and others.

14.1. Ingest And Inspect Data

The foundation of any good data Pipeline is the ability to reliably onboard, simulate, and validate your data. Before applying complex transformations, you must ensure that your data is correctly shaped and that your processing logic behaves as intended under real-world conditions.

Cribl Edge has various tools that allow you to move from initial setup to a verified, production-ready state:

- Onboard Data
- Use Datagens to Simulate Live Data
- Create and Share Data Samples
- Data Sample Use Cases
    - Ingest-Time Sampling
    - Sample Logs
    - Access Logs: Apache, ELB, CDN, S3
    - Firewall Logs: VPC Flow Logs, Cisco ASA

14.1.1. Onboard Data

Onboarding data into Cribl Edge can vary in complexity, depending on your organization's needs, requirements, and constraints. Proper onboarding from all Sources is key to system performance, troubleshooting, and ultimately the quality of data and decisions both in Cribl Edge and in downstream Destinations.

General Onboarding Steps

Typically, a data onboarding process revolves around these steps, both before and after turning on the Source:

- Create configuration settings.
- Verify that settings do the right thing.
- Iterate.

Below, we break down individual steps.

Before Turning On the Source

Cribl recommends that you take the following steps to verify and tune incoming data, before it starts flowing.

Preview Sample Data

Use a sample of your real data in Data Preview. Sample data can come from a sample Source file that you upload or paste into Cribl Edge.

You can also obtain sample data in a live data capture from a Source. One way to do this **before** going to production is to configure your Source with a **devnull** Pipeline (which just drops all events) as a pre-processing Pipeline. Then, let data flow in for just long enough to capture a sufficient sample.



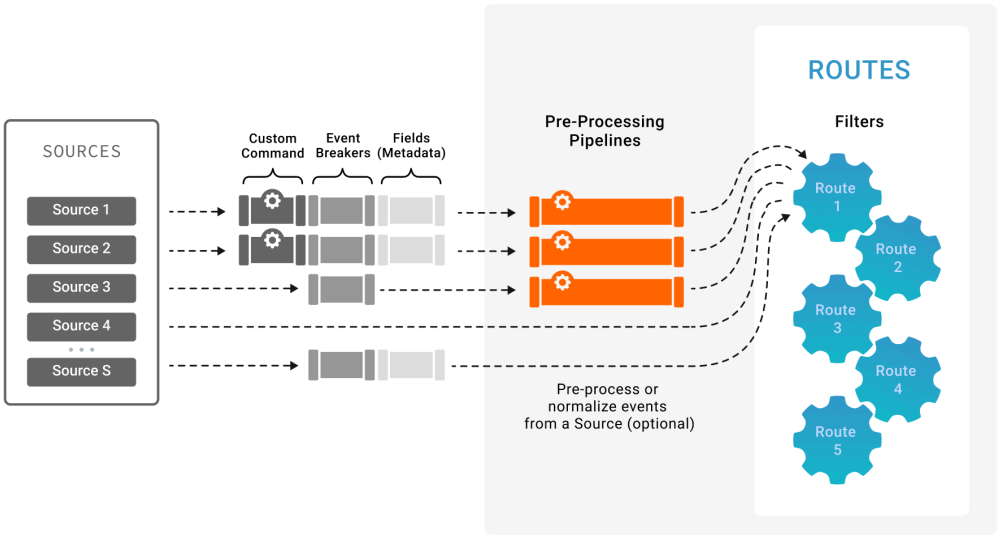
Very Large Integer Values

Cribl Edge's JavaScript implementation can safely represent integers only up to the Number.MAX_SAFE_INTEGER constant of about 9 quadrillion (precisely, {2^53}-1). Data Preview will round down any integer larger than this, and trailing 0's might indicate such rounding down.

Check the Processing Order

While events can be processed almost arbitrarily by Functions in Cribl Edge Pipelines, make sure you understand the event processing order. This is very important, as it tells you exactly where certain processing

steps occur. For instance, as we'll see just below, quite a few steps can be accomplished at the Source level, before data even hits Cribl Edge Routes.



Source-level processing options

Custom Command

Where supported, data streams will be handled by **custom commands**. These are external system commands that can (optionally) be used to pre-process the data. You can specify any command, script, etc., that consumes via `stdin` and outputs via `stdout`.

Verify that such commands are doing what's expected, as they are the very **first** in a series of processing steps.

Event Breakers

Next, data streams are handled by Event Breakers, which:

- Convert data streams into discrete events.
- Extract and assign timestamps to each event.

If the resulting events do not look correct, feel free to use **non-default** breaking rules and timestamp recognition patterns. Downstream, you can use the Auto Timestamp Function to modify `_time` as needed, if timestamps were not recognized properly. Examples of such errors are:

- Timestamps too far out in the future or past
- Wrong timezone.
- Incorrect timestamp is selected from multiple timestamps present in the event.

Fields

Next, events can be enriched with Fields . This is where you'd add static or dynamic fields to all events delivered by a particular Source.

Pre-Processing Pipeline

Next, you can optionally configure a pre-processing Pipeline on a particular Source. This is extremely useful in these cases:

- Drop non-useful events as early as possible (so as to save on CPU processing).
- Normalize events from this Source to conform a certain shape or structure.
- Fix/touch up events accordingly. E.g., if event breakers assigned the wrong timestamp, this is the best place to use the Auto Timestamp Function to adjust `_time`.

We Can't Say This Enough

Verify, verify, verify, data integrity before turning on the Source.

After Turning On the Source

Use data Destinations to verify that certain metrics of interest are accurate. This will depend significantly on the capabilities of each Destination, but here's a basic checklist of things to ensure:

- Timestamps are correct.
- All necessary fields are assigned to events.
- All expected events show up correctly. (E.g., if a Drop or Suppress Function was configured, ensure that it's not dropping unintended events.)
- Throughput - both in bytes and in events per second (EPS) - is what's expected, or is within a certain tolerance.

Iterate

Iterate on the steps above as necessary. E.g., adjust fields values and timestamps as needed.



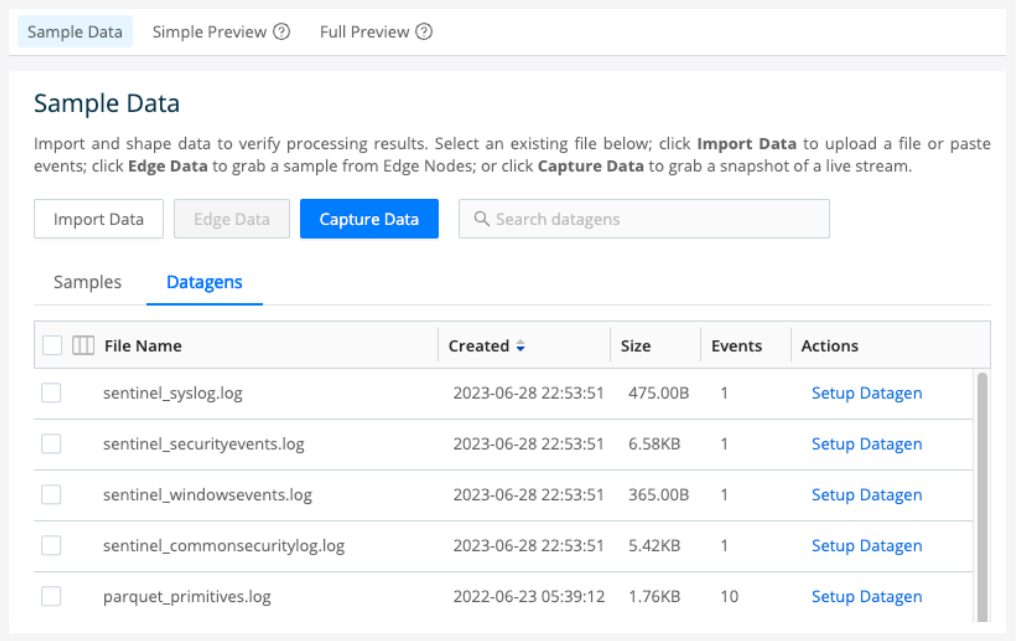
Remember that there is almost always a workaround. Any arbitrary event transformation that you need is likely just a Function or two away.

14.1.2. Use Datagens To Simulate Live Data

Data generators for testing and troubleshooting

Cribl Edge's datagens feature enables you to generate sample data for the purposes of troubleshooting Routes, Pipelines, Functions, and general connectivity.

Several datagen template files ship with the product, out of the box. You can create others from sample files or live captures.



Preview pane - add samples via paste, attach/upload file, or live capture

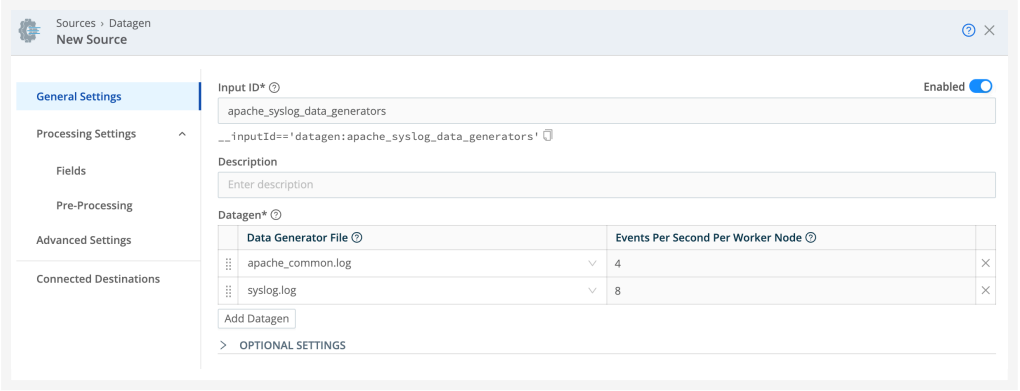
As outlined in the following tutorial: Once you've created a template, you can configure a Datagen Source to use the template to generate real-time data at a given EPS (events per second) rate.

Enable a Datagen

To see how datagens work, start by enabling a pair of Cribl Edge's out-of-the-box generators:

Navigate to **Sources** > **Datagens** and select **Add Source**.

After entering a unique ID and an optional description, select a data generator file (such as `apache_common.log`) and set it at `4` **Events Per Second Per Worker Node**. Select **Add Datagen** to select another data generator file (such as `syslog.log`) and set it at `8` **Events Per Second Per Worker Node**. Select **Save**.



Selecting datagen files and event rates

In the sidebar, select **Monitoring**, select **Data** > **Sources**, and search for `datagen` to confirm that the Source is generating data.

Create a Datagen Template from a Sample File

To convert a sample into a template:

From the **Data Routes** or **Pipelines** page, select **Import Data** in the right preview pane, and paste a sample like the AWS VPC Flow logs below:

Sample VPC Flow Logs

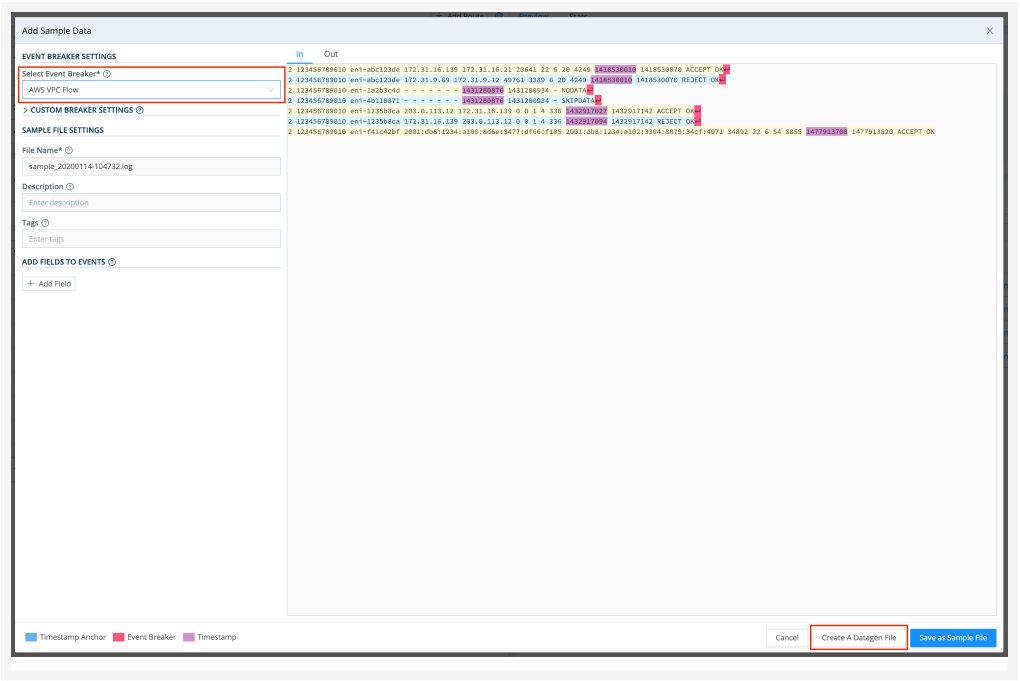
```
2 123456789010 eni-abc123de 172.31.16.139 172.31.16.21 20641 22 6 20 4249 1418530010
2 123456789010 eni-abc123de 172.31.9.69 172.31.9.12 49761 3389 6 20 4249 1418530010 1
2 123456789010 eni-1a2b3c4d - - - - - - - 1431280876 1431280934 - NODATA
2 123456789010 eni-4b118871 - - - - - - - 1431280876 1431280934 - SKIPDATA
2 123456789010 eni-1235b8ca 203.0.113.12 172.31.16.139 0 0 1 4 336 1432917027 1432917
2 123456789010 eni-1235b8ca 172.31.16.139 203.0.113.12 0 0 1 4 336 1432917094 1432917
2 123456789010 eni-f41c42bf 2001:db8:1234:a100:8d6e:3477:df66:f105 2001:db8:1234:a102
```

Copy

From the **Select Event Breaker** drop-down, select **AWS VPC Flow** to ensure that:

- The pasted text gets broken properly into individual events (notice the Event Breaker on newlines).
- Timestamps are extracted correctly (text highlighted purple below).

Once you've verified these results, select **Create a Datagen File**.

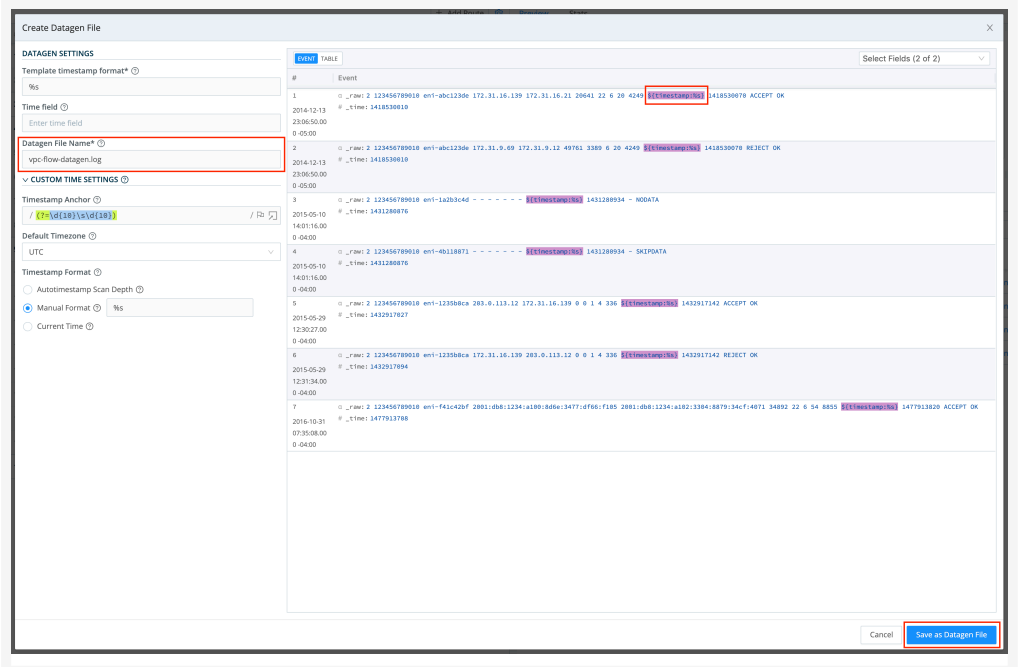


Creating a datagen template

On the resulting **Create Datagen File** screen:

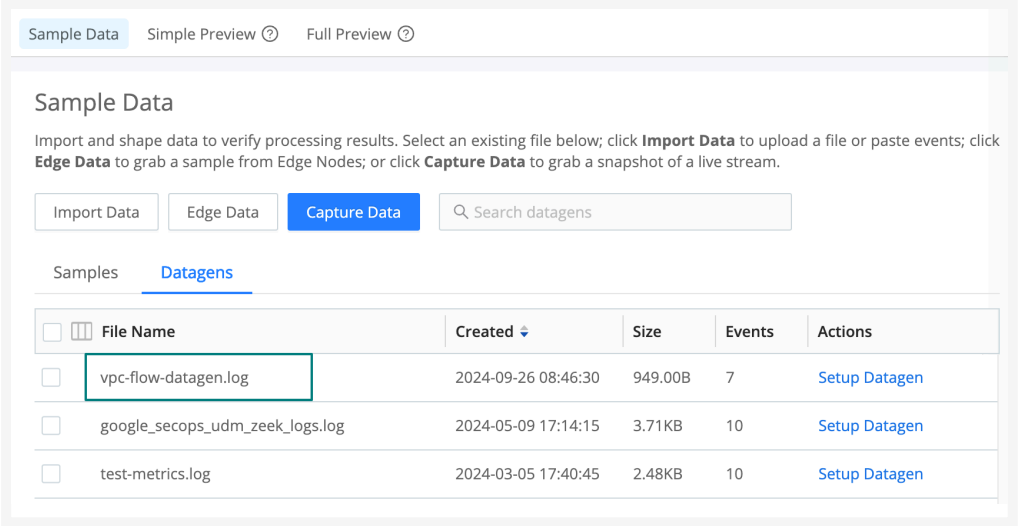
- Enter a file name, such as `vpc-flow-datagen.log`
- Ensure that the timestamp template format is correct: `${timestamp: %s}`
  `${timestamp: <format>}` is a template that the datagen engine uses to insert the current time - in each newly generated event - using the given format. In this case, `%s` is the desired `strftime` format for the timestamp (the epoch).

Once you've verified these results, click **Save as Datagen File**.



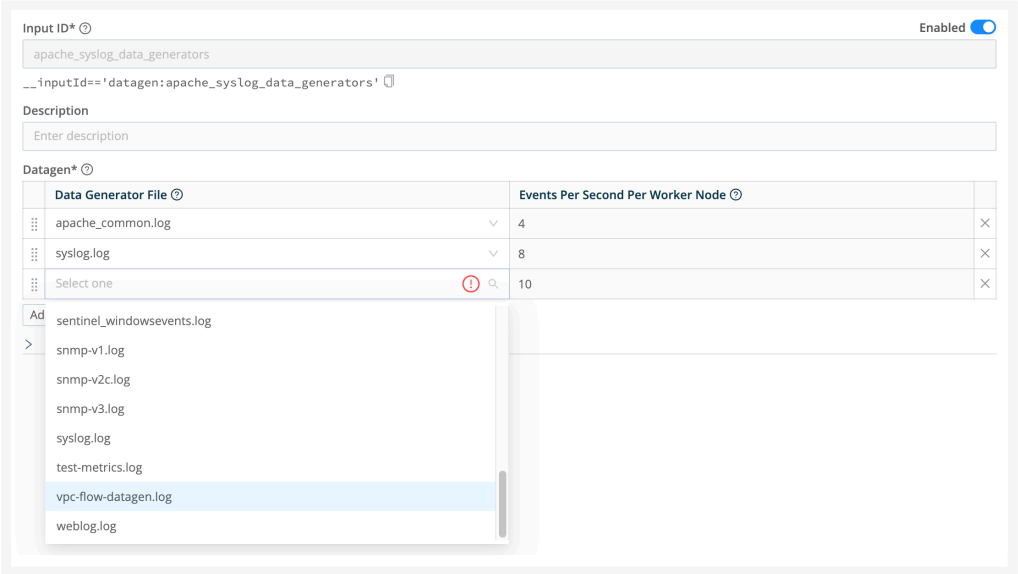
Saving a named datagen template

To confirm that the datagen file has been created, select the **Datagens** tab in the preview pane.



Verifying datagen file creation

Now, to start using your newly created datagen file, go back to **Sources** > **Datagens**. Add it using the drop-down shown below.

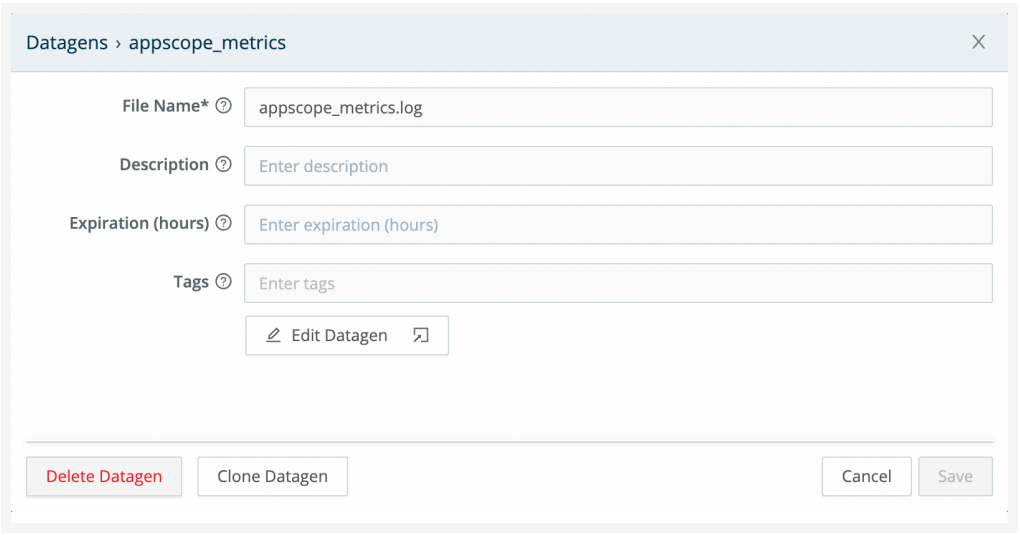


Adding new template file to datagens Source

Modify a Datagen

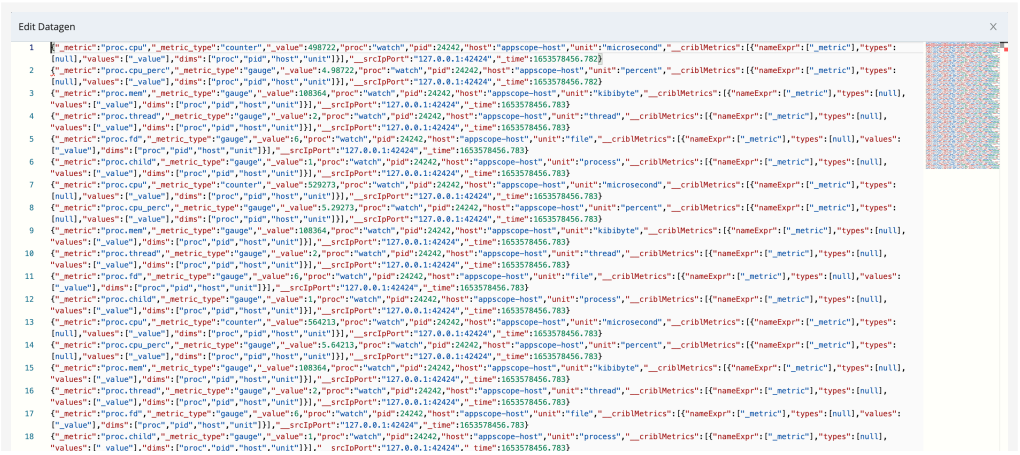
1. In the right Preview pane, select the **Datagens** tab.
2. Hover over the file name you want to modify. This displays an edit (pencil) button to its left.

3. Select that button to open the modal shown below. It provides options to edit the datagen, clone it, delete it, or modify its metadata (**File name**, **Description**, **Expiration time**, and **Tags**).



Options for modifying a datagen

4. To make changes to the datagen, select the modal's **Edit Datagen** button. This opens the **Edit Datagen** modal shown below, exposing the raw data that this datagen uses to generate events.
5. Edit the raw data as desired.
6. Select **Update Datagen** to resave the modified datagen, or select **Save as New Datagen** to give the modified version a new name.



14.1.3. Create And Share Data Samples

A data sample (also called a data capture) is a small, representative subset of live or stored data from a given data stream. Sample files are essential tools for:

- Ensuring that proper data is flowing between the right Sources and Destinations.
- Confirming that data is routed correctly.
- Testing and validating Routes, Pipelines, and Functions to ensure they process data as expected.

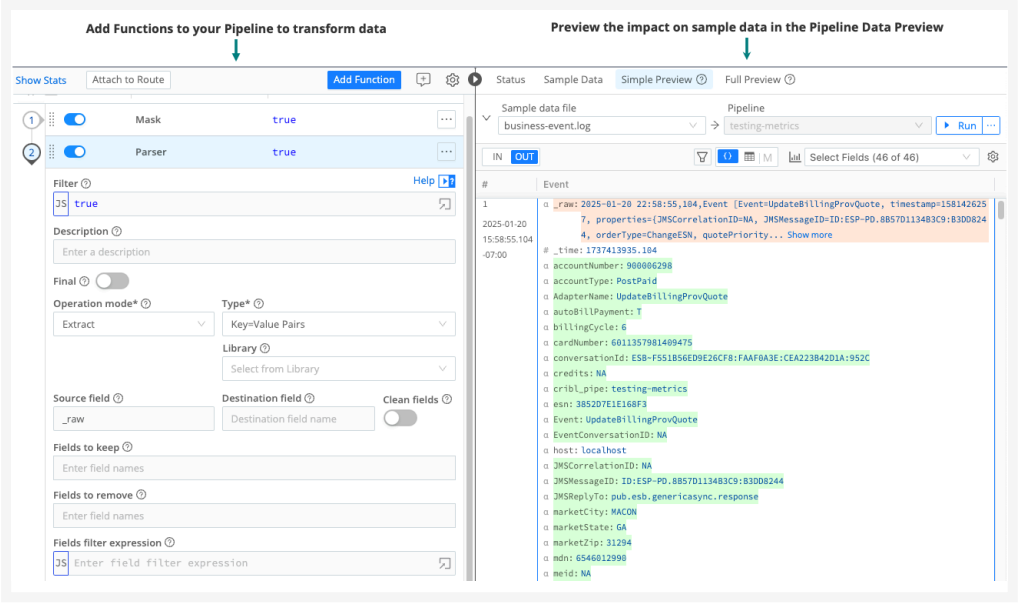
Access and Manage Data Samples

Whether you are capturing live data samples for immediate testing or managing a shared library for collaboration, you can work with data samples through a few main workflows:

Workflow	Description
Data Preview	Capture sample data, then test and validate Pipeline logic in real-time using the Data Preview tool as you build Routes and Pipelines to see your data processing logic in action.
Knowledge Objects	Save samples as Knowledge objects in a central location to manage and reuse them across Routes, Pipelines, Event Breakers, and datagens. Samples you create here can also be used in the Data Preview tool.
Datagens	Generate a stream of realistic sample log data to help you simulate live data. Build custom datagens (data generators) from your own sample files.

Data Preview

The primary way that you will use sample files is with the Data Preview tool for Pipelines and Routes. The Data Preview tool works by processing a set of sample events and passing them through the Pipeline or Route. Then, it displays the inbound and outbound results in a dedicated pane. Whenever you modify, add, or remove a Function, the output updates instantly to reflect your changes. This is helpful for validating your data processing logic in real-time to assess whether the logic works as you expected.



Add Functions to your Pipeline to transform data, then preview the impact on sample data in Data Preview

See Validate Pipeline Logic Using Data Preview for information about accessing and using the Data Preview tool.

Knowledge Objects

You can create and save sample data files as Knowledge objects at either the Fleet level or the Pack level. The advantage of creating samples as Knowledge objects is that these objects can be accessed across your Fleet by Routes, Pipelines, Event Breakers, and datagens as needed.

When working with sample files as Knowledge objects, be aware that sample file visibility is context-dependent:

- Fleet samples are only visible within that specific Fleet.
- Pack samples are only visible within that specific Pack.

To use a sample in a different context, you must manually copy it from the Fleet to the Pack (or the Pack to the Fleet). See Copy and Share Sample Files for specific instructions.

To access sample files in the Knowledge library at the Fleet level:

1. Select **Fleets** from the sidebar, and choose a Fleet.
2. On the **Fleets** submenu, select **More**, then **Knowledge**, then **Sample Files**.

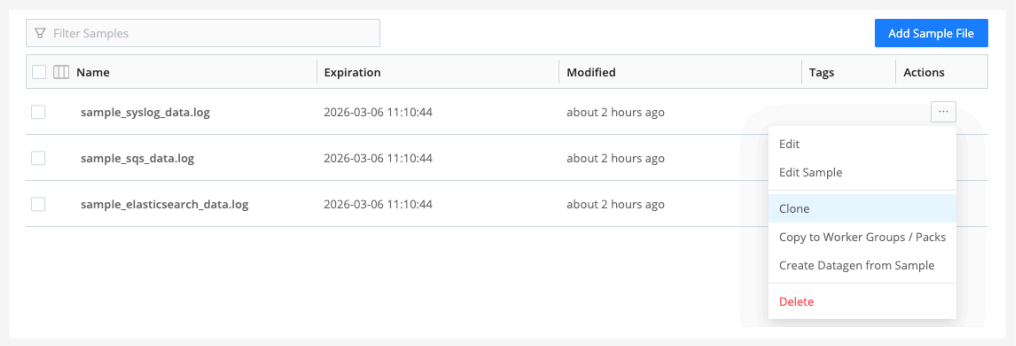
To access sample files in the Knowledge library from inside a Pack:

1. Ensure you are in the correct Pack.
2. Select **Knowledge** from the top bar, then **Sample Files**.

The Knowledge library lists any sample files you have captured. From the list view, you add new files, edit samples, clone, or copy sample files to other Fleets or Packs. See Copy and Share Sample Files for specific instructions.

Cross-copying samples between Worker Groups and Fleets is not supported.

You can also build a custom datagen (data generator) from these sample files to simulate live data. See the next section about datagens for more information.



View and manage sample files in the Knowledge objects library

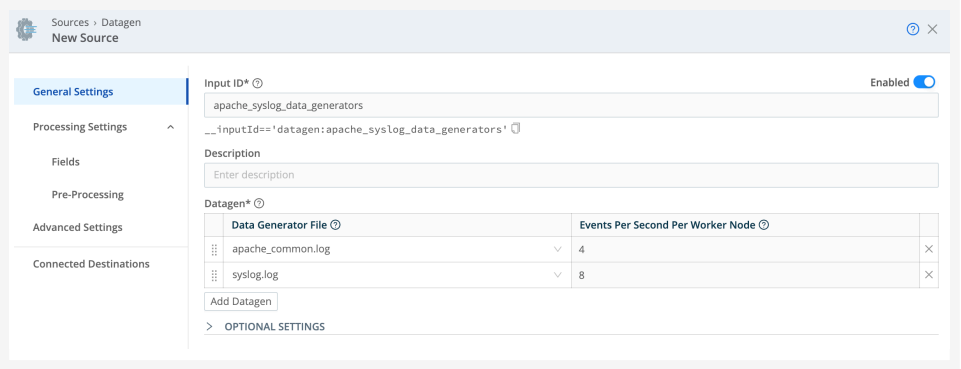
Datagens

Datagens (data generators) allow you to simulate live data by replaying sample log data. You can then use this live data to test and validate your Routes, Pipelines, and Functions in a sandbox environment before applying changes to production data.

Cribl provides several datagens that you can use out-of-the-box to simulate live data. You can also transform your own custom sample files into active generators. See Use Datagens to Simulate Live Data for more information.

If you create a sample file in the Knowledge object library, you can use this file to build a custom datagen. To create the datagen from a sample file:

1. Ensure you are in the correct context (either in a Fleet or a Pack). Then access the Knowledge objects library in that context.
2. Next to the sample file you want to copy or share, select **Create Datagen from Samples** from the **Actions** (⋯) menu.
3. In the modal, update the datagen settings as needed. See Use Datagens to Simulate Live Data for more information.



Simulate live data with a data generator

4. Select **Save as Datagen File**.

The new datagen is automatically added to your list of Sources for that Fleet or Pack. It will share the same name as the original sample file.

Add Sample Data

To validate your data, you first need sample data to work with. Some methods of getting sample data can only work with the Data Preview tool. If you are creating a data sample using the Knowledge object library, you can only capture sample data live from a Source or from a datagen.

You can get sample data using a few methods:

Method	Description	Requirements and Limitations
Capture new sample data	Capture live data from multiple incoming data streams to use for validating data routing and processing logic.	• Data captures can only work with events. • Data captures do not require event breaking. • To capture live data, you must have Edge Nodes registered to the Fleet for which you're viewing events.
Capture live data from a single Source or Destination	Use this method if you only need to capture data from one Source or Destination to build your data sample.	• To capture live data from a single enabled Source or Destination, it's fastest to use the Sources or Destinations pages instead of the Preview pane. • To capture live data, you must have Edge Nodes registered to the Fleet for which you're viewing events.
Upload a sample data file	Use this method to upload sample data file from another	• Available for Data Preview only.

Method	Description	Requirements and Limitations
	Cribl deployment or an external data source.	• The **Import Data** options require content that is broken into events, which means it needs Event Breakers.
Copy data from a clipboard	Use this method to copy and paste sample data from your computer's internal clipboard.	• Available for Data Preview only. • The **Import Data** options require content that is broken into events, which means it needs Event Breakers.
Import data from Cribl Edge Nodes	Use this method to import data from Cribl Edge.	• Available for Data Preview only. • You must have at least one working Edge Node.

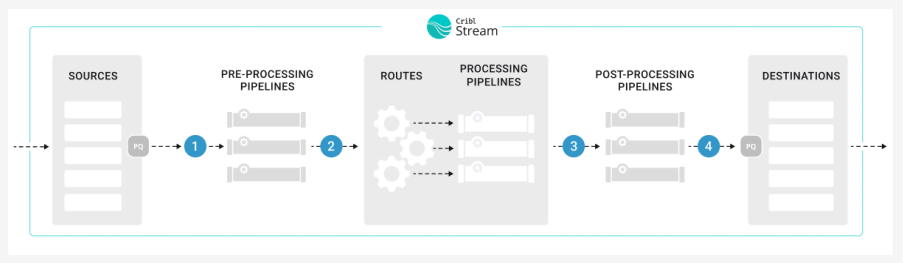
Capture New Sample Data

The **Capture Data** mode does not require event breaking. To capture live data, you must have Edge Nodes registered to the Fleet for which you're viewing events. You can view registered Edge Nodes from the **Status** tab in the Source.

To capture new sample data:

1. Ensure you are in the correct context (such as the Fleet or inside a Pack). To navigate to the Knowledge library:
    - **From a Fleet:** Select **Fleets** from the sidebar, and choose a Fleet. On the **Fleets** submenu, select **More**, then **Knowledge**, then **Sample Files**.
    - **From a Pack:** Select **Knowledge** from the top bar, then **Sample Files**.
2. Select **Add Sample File**.
3. Select **Capture** and modify the settings as needed:
    - **Capture time (sec):** The length of the time you want to sample in seconds. Defaults to 10 seconds.
    - **Capture up to N events:** The maximum number of events to capture. Defaults to 10.
    - **Where to capture:** Select the stage in the data flow where you want to capture events. Options include:
        - Before pre-processing Pipeline (labeled as `1` in the following diagram)
        - Before the Routes (2)
        - Before post-processing Pipeline (3)

- Before the Destination (4)



Options for where to capture the data

- **Initial Worker sample size:** Set a limit to the number of Edge Nodes that are included in the capture. Defaults to 50 Nodes. Set it to `0` to make it unlimited.
- **Interval (sec):** Specify how long the system should wait for data before increasing the capture sample size. Must be at least 1 second. Defaults to 2 seconds.

4. Select **Start** and the system will start capturing data.

If the capture fails, check that you have committed and deployed all changes. Also check the **Filter** setting for the data capture to ensure it is not filtering out all events.

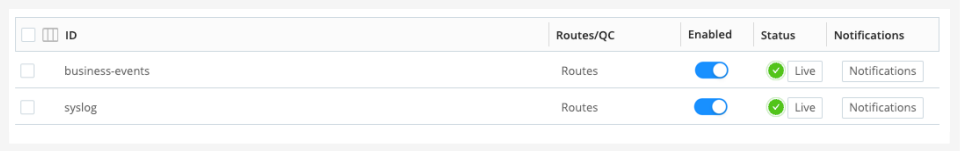
5. Select **Save as Sample File**.

Verify that the new sample file appears in the Data Preview pane or in the Knowledge objects library.

Capture Live Data from a Single Source or Destination

To capture data from a single enabled Source or Destination, it's fastest to use the Sources or Destinations pages instead of the Preview pane:

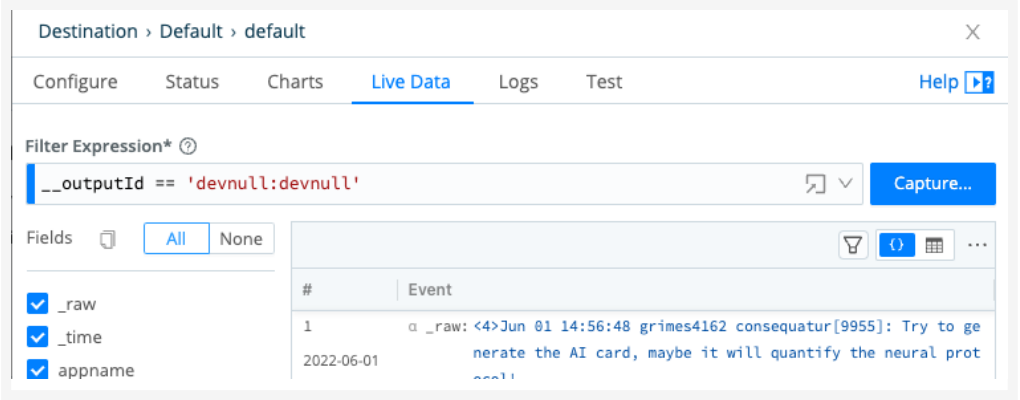
1. In the sidebar, select **Fleets**, then select a Fleet. On the **Fleets** submenu, select **More**, then **Sources** or **Destinations**.
2. Select **Live** on the Source or Destination configuration row to initiate a live capture.



Live button on a Source

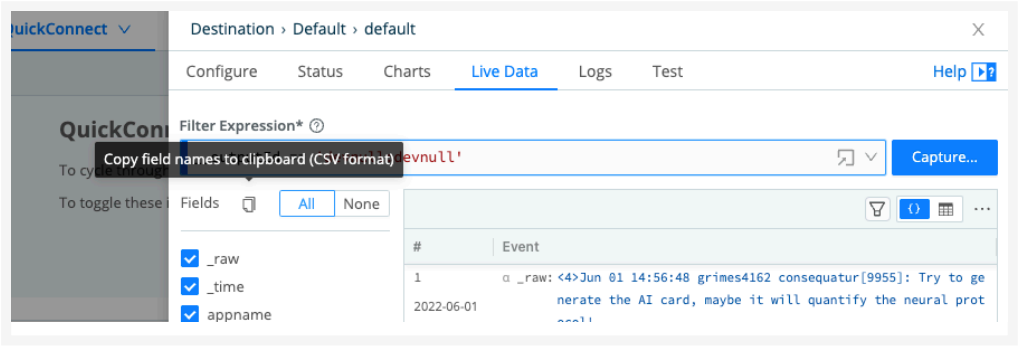
3. Select **Capture**. Accept or modify the default settings as needed and select **Start**.
4. When the modal finishes populating with events, select **Save as Sample File**. In the sample file settings, configure the sample's metadata. See Sample Data Settings for detailed information about these settings.

Alternatively, you can start an immediate capture from within an enabled Source's or Destination's configuration modal, by selecting the modal's **Live Data** tab.



The Live Data tab in the Destination modal

You can also use the **Fields** selector in the **Live Data** is a Copy button, which enables you to copy the field names to the clipboard in CSV format. The **Logs** tab also provides this copy button.

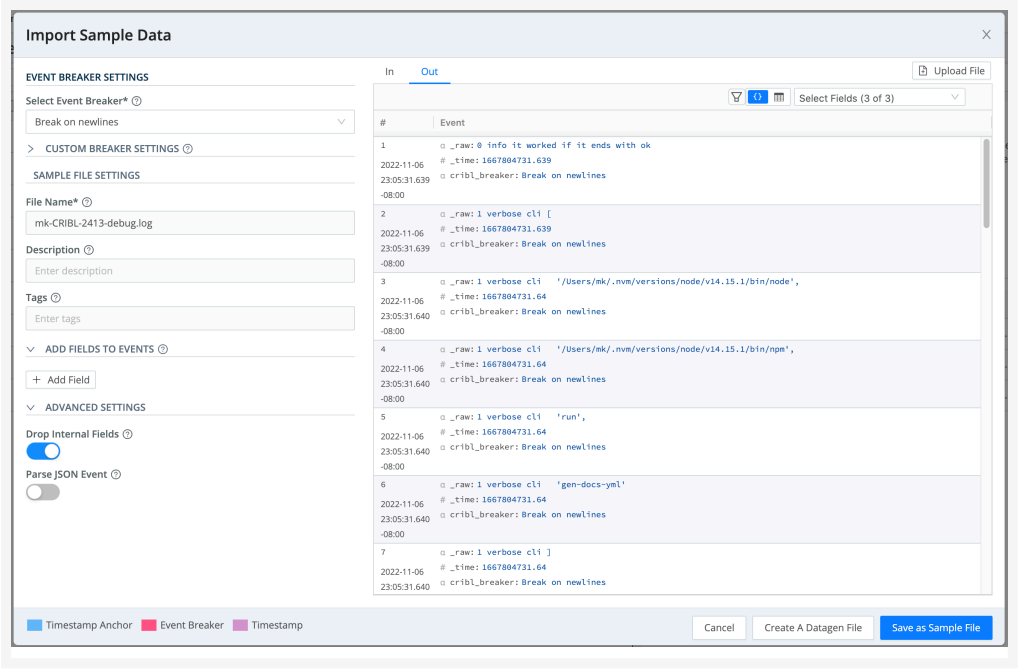


The Copy Fields Icon in the Live Data tab

Upload a Sample Data File

To upload a sample data file:

1. In the sidebar, select **Fleets**, then select a Fleet. On the **Fleets** submenu, select **More**, then **Pipelines**.
2. Select the **Sample Data** tab in the **Data Preview** pane. Select **Import Data**.
3. In the **Import Sample Data** modal, drag and drop your sample file or select **Upload** to navigate to the file on your computer.
4. Select an appropriate option from the **Event Breaker Settings** menu.
5. Configure the sample's metadata. See Sample Data Settings for detailed information about these settings.
6. Add or drop fields as needed.



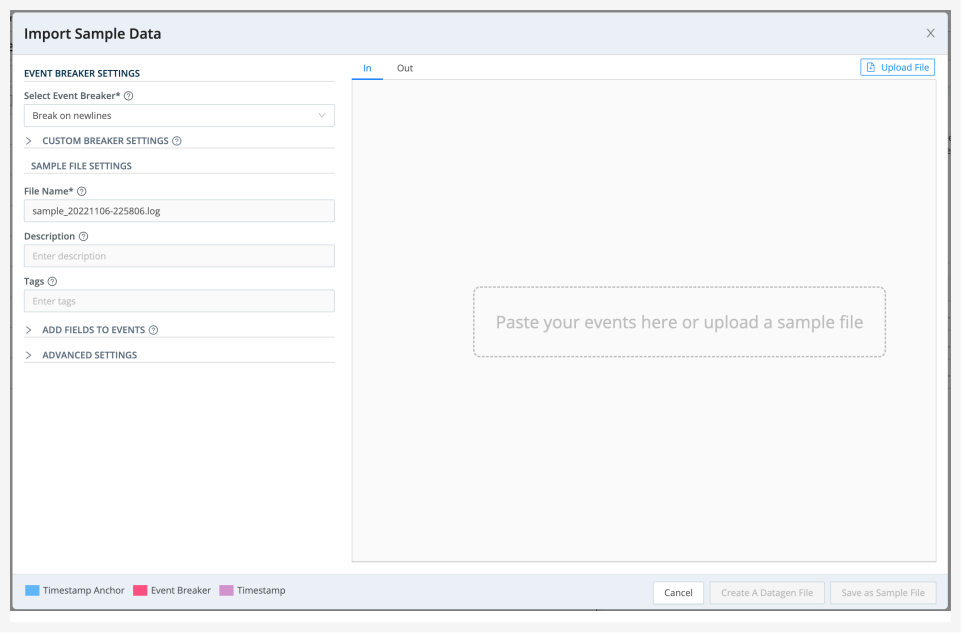
Example of imported data

If you notice imported events are fragmented, it could be because Cribl Edge has added a `__timeoutFlush` internal field to them. See the troubleshooting section about Imported Data Events Are Fragmented for more information.

Copy Data From a Clipboard

To copy sample data from a clipboard:

1. In the sidebar, select **Fleets**, then select a Fleet. On the **Fleets** submenu, select **More**, then **Pipelines**.
2. Select the **Sample Data** tab in the **Data Preview** pane. Select **Import Data**.
3. In the **Import Sample Data** modal, paste data from your computer clipboard into the provided field.



Import Sample Data modal

4. Select an appropriate option from the **Event Breaker Settings** menu.
5. Configure the sample's metadata. See Sample Data Settings for detailed information about these settings.
6. Add or drop fields as needed.

If you notice imported events are fragmented, it could be because Cribl Edge has added a `__timeoutFlush` internal field to them. See the troubleshooting section about Imported Data Events Are Fragmented for more information.

Import Data from Cribl Edge Nodes

To import data from Cribl Edge Nodes, you must have at least one working Edge Node. To upload data from a file on an Edge Node:

1. In the sidebar, select **Fleets**, then select a Fleet. On the **Fleets** submenu, select **More**, then **Pipelines**.
2. Select the **Sample Data** tab in the **Data Preview** pane. Select **Import Data**.
3. In the **Import Sample Data** modal, select **Edge Data** and navigate to the Edge Node that has the stored file.
4. In **Manual** mode, use the available filters to narrow the results:
    - **Path**: The location from which to discover files.
    - **Allowlist**: Supports wildcard syntax and supports the exclamation mark (!) for negation.
    - **Max depth**: Sets which layers of files to return highlighted in bold typeface. By default, this field is empty, which implicitly specifies `0`. This default boldfaces only the top-level files within the **Path**.

5. Once you find the file you want, select its name to add its contents to the **Add Sample Data** modal.
6. Select an appropriate option from the **Event Breaker Settings** menu.
7. Configure the sample's metadata. See Sample Data Settings for detailed information about these settings.
8. Add or drop fields as needed.

Modify Sample Files

To modify sample files that were captured and stored as Knowledge objects, you can edit samples by accessing the list of sample files in the Knowledge objects library. From there, select the **Edit Sample** option from the **Actions** (...) menu.

To modify existing sample files in the Data Preview tool in Routes or Pipelines:

1. In the right Preview pane, select the **Samples** tab.
2. Hover over the file name you want to modify to show an edit (pencil) button next to the file name.
3. Select that button to open the modal shown below. It provides options to edit, clone, or delete the sample, save it as a datagen Source, or modify its metadata. See Sample Data Settings for detailed information about these settings.
4. To make changes to the sample, select the modal's **Edit Sample** button. This opens the **Edit Sample** modal shown below, exposing the sample's raw data.
5. Edit the raw data as desired.
6. Select **Update Sample** to resave the modified sample, or select **Save as New Sample** to give the modified version a new name.



Editing a sample file

Copy and Share Sample Files

To clone a sample file in the Data Preview tool, follow the same steps to Modify Sample Files.



Sample files cannot be copied from a Worker Group to a Fleet.

To copy sample files that were captured and stored as Knowledge objects:

1. Select **Fleets** from the sidebar, and choose a Fleet. On the **Fleets** submenu, select **More**, then **Knowledge**, then **Sample Files**.
2. Next to the sample file you want to copy or share, select **Copy to Fleets / Packs** option from the **Actions** (...) menu.
3. In the **Copy Sample** modal, define your target:
    - **Fleet:** Type or select the target Fleet. Leave blank if you only want to copy the file to a Pack.
    - **Packs:** Type or select the target Pack. Leave blank if you only want to copy the file to a Fleet.
4. Select **Copy Sample**.

A confirmation message will appear upon a successful copy.

If you want to share sample files across your Cribl deployment, add them to a Pack and then copy the Pack to other Fleets. From there, you can copy it to the target Fleet. See Copy a Pack for specific instructions.

Sample Data Settings

These metadata settings are available for all data samples:

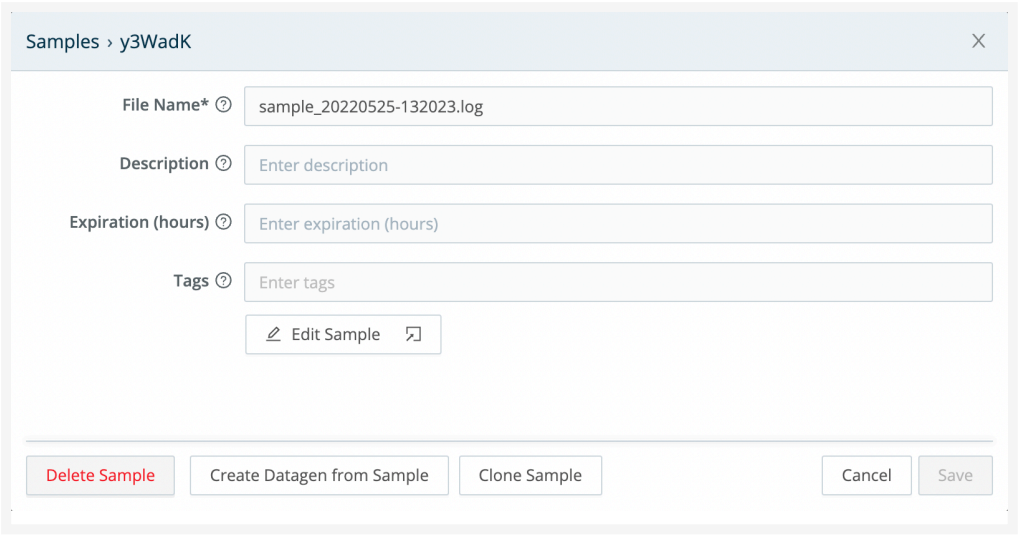
- **File name**: The display name for the sample file. Choose a name that makes the sample easy to recognize in the **Sample Data** and **Samples** tabs.
- **Description**: An optional, free-form description that explains what the sample represents (source system, time range, scenario, and so on).
- **Expiration time**: A time-to-live (TTL), in hours, for the sample file. Each time you use the sample, for example in Data Preview, Pipeline testing, or when creating a datagen, the TTL resets. When a sample reaches its expiration without being used, Cribl Edge silently removes the sample and its backing file. Leave this field blank if you do not want the sample to expire automatically.



In deployments that use Git-backed configuration bundles, expiration only affects the live configuration on disk. If you commit while the sample exists, that commit continues to contain the sample. When the

sample later expires, its removal is just another change that you can commit in a later revision. If the sample expires before you commit, it is cleaned up without ever being recorded in Git.

- **Tags**: Optional labels that you can use to organize and search for samples (for example, by source, environment, or use case).



Options for modifying a sample

Troubleshoot Sample Data

This section describes some strategies for managing potential issues with sample data.

Integer Values Are Too Large

The JavaScript implementation in Cribl Edge can safely represent integers only up to the Number.MAX_SAFE_INTEGER constant of about 9 quadrillion (precisely, {2^53}-1). Data Preview will round down any integer larger than this, and trailing zeroes might indicate such rounding.

Imported Data Events Are Fragmented

The **Import Data** options work with content that needs to be broken into events, meaning it needs Event Breakers.

Cribl Event Breakers are regular expressions that tell Cribl Edge how to break the file or pasted content into events. Breaking occurs at the **start** of the match. Cribl Edge ships with several common breaker patterns out of the box, but you can also configure custom breakers. The UI here is interactive, and you can iterate until you find the exact pattern.

If you notice fragmented events, check whether Cribl Edge has added a `__timeoutFlush` internal field to them. The presence of this diagnostic field indicates that Cribl Edge flushed the events because the Event

Breaker buffer timed out while processing them. These timeouts can be due to large incoming events, backpressure, or other causes. See Event Model: Internal Fields for more information.

14.2. Event Data Structure And Flow

This section provides a complete reference for how Cribl Edge receives, formats, and segments raw data into structured events.

Core Concepts

- Event Model: Learn about the internal structure of a Cribl Edge event.
- Event Processing Order: A step-by-step breakdown of the evaluation order of operations for incoming data.

Event Breaking and Parsing

- Event Breakers: Explains how event breaking works in Cribl Edge and how to apply them to your raw data to turn it into structured data.
- Event Breaker Types: Learn about the configuration and use cases for each Event Breaker type, allowing you to choose the most efficient parser for your specific log format.

14.2.1. Event Model

Learn how Cribl Edge represents and captures events.

All data processing in Cribl Edge is based on discrete data entities commonly known as **events**. An event is defined as a collection of key-value pairs (fields). Some Sources deliver events directly, while others might deliver bytestreams that need to be broken up by Event Breakers. Events travel from a Source through Pipelines' Functions, and on to Destinations.

The internal representation of a Cribl Edge event is as follows:

Event Model

```
{
  "_raw": "<body of non-JSON parse-able event>",
  "_time": "<timestamp in Unix time>",
  "__inputId": "<Id/Name of Source that delivered the event>",
  "__other1": "<Internal field1>",
  "__other2": "<Internal field2>",
  "__otherN": "<Internal fieldN>",
  "key1": "<value1>",
  "key2": "<value2>",
  "keyN": "<valueN>",
  "...": "..."
}
```

Copy

Some notes about these representative fields:

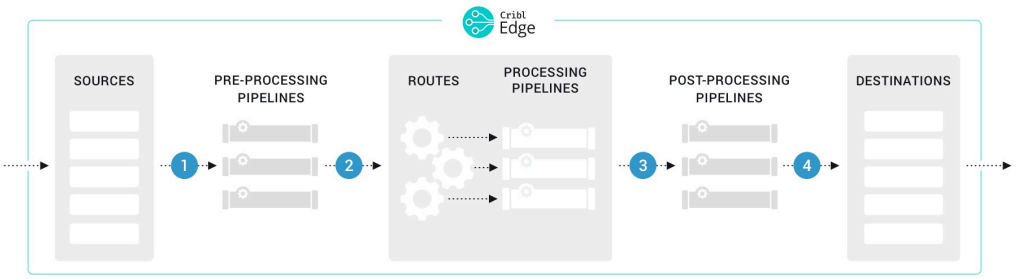
- Fields that start with a double-underscore are known as **internal fields**, and each Source can add one or many to each event. For example, Syslog adds both a `__inputId` and a `__srcIpPort` field. Internal fields are used only within Cribl Edge, so they are serialized only to Cribl internal Destinations.
- Upon arrival from a Source, if an event cannot be JSON-parsed, all of its content will be assigned to `_raw`.
- If a timestamp is not configured to be extracted, the current time (in Unix time) will be assigned to `_time`.
- Cribl reserves the right to change all internal fields that are not documented under Sources or Destinations.

Using Capture

One way to see what an event looks like as it travels through the system is to use the **Capture** feature. While in Data Preview (right pane):

1. Click **Start a Capture**.
2. In the resulting modal, enter a **Filter expression** to narrow down the events of interest.
3. Click **Capture...** and (optionally) change the default Time and/or Event limits.
4. Select the desired **Where to capture** option. There are four options:

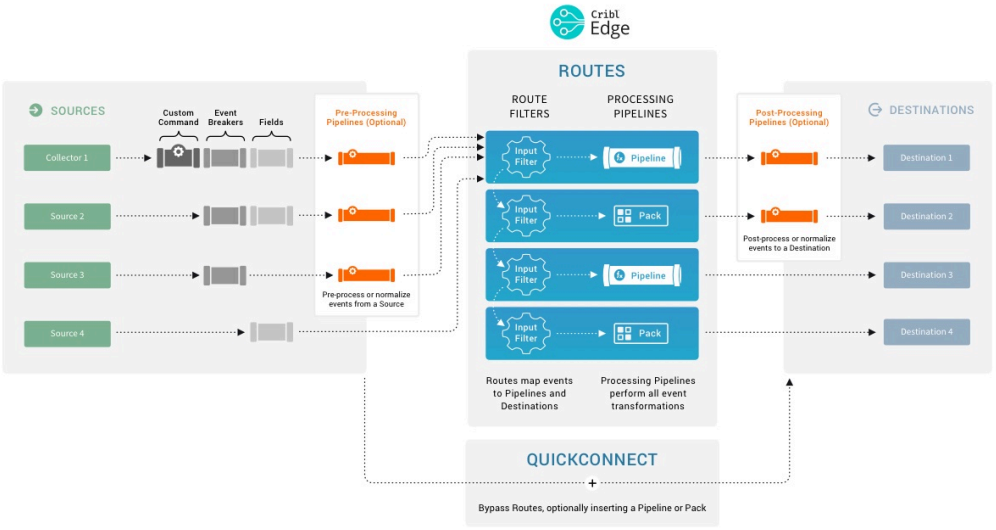
- **1. Before the pre-processing Pipeline** - Capture events right after they're delivered by the respective Input.
- **2. Before the Routes** - Capture events right after the pre-processing Pipeline, before they go down the Routes.
- **3. Before the post-processing Pipeline** - Capture events right after the Processing Pipeline that actually handled them, before any post-processing Pipeline.
- **4. Before the Destination** - Capture events right after the post-processing Pipeline, before they go out to the configured Destination.



14.2.2. Event Processing Order

See how data flows through Cribl Edge.

The expanded schematic below shows how all events in the Cribl Edge ecosystem are processed linearly, from left to right.



Cribl Edge in great detail

Here are the stages of event processing:

1. Sources: Data arrives from your choice of external providers. (Cribl Edge supports Splunk, HTTP/S, Elastic Beats, Amazon Kinesis/S3/SQS, Kafka, TCP raw or JSON, and many others.)
2. Custom command: Optionally (where available), you can pass this input's data to an external command before the data continues downstream. This external command will consume the data via `stdin`, will process it and send its output via `stdout`.
3. Event Breakers can, optionally, break up incoming bytestreams into discrete events. (Note that because Event Breakers precede tokens, Breakers cannot see or act on tokens.)
4. Fields/Metadata: Optionally, you can add these enrichments to each incoming event. You add fields by specifying key-value pairs, per Source, in a format similar to Cribl Edge's Eval Function. Each key defines a field name, and each value is a JavaScript expression (or constant) used to compute the field's value.
5. Pre-processing Pipeline: Optionally, you can use a single Pipeline to condition (normalize) data from this input before the data reaches the Routes.
6. Routes map incoming events to Processing Pipelines and Destinations. A Route can accept data from multiple Sources, but each Route can be associated with only one Pipeline and one Destination.

7. Processing Pipelines perform all event transformations. Within a Pipeline, you define these transformations as a linear series of Functions. A Function is an atomic piece of JavaScript code invoked on each event.
8. Post-processing Pipeline: Optionally, you can append a Pipeline to condition (normalize) data from each Processing Pipeline before the data reaches its Destination.
9. Destinations: Each Route/Pipeline combination forwards processed data to your choice of streaming or storage Destination. (Cribl Edge supports Splunk, Syslog, Elastic, Kafka/Confluent, Amazon S3, Filesystem/NFS, and many other options.)

Troubleshoot event processing in Cribl Stream from right to left. Start at the Destination, and check for block status from the Destination back to the Source. If one Worker Group is sending to another, the right-most destination will be in the receiving Worker Group.



Pipelines Everywhere

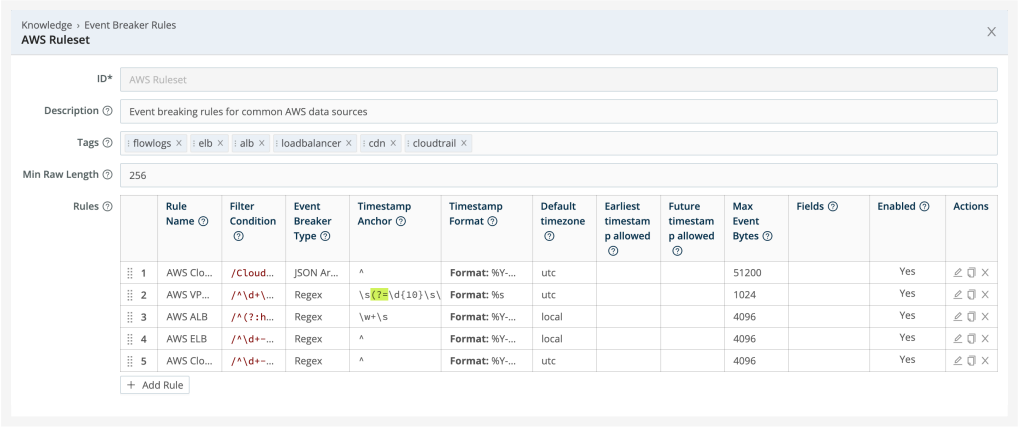
All Pipelines have the same basic internal structure - they're a series of Functions. The three Pipeline types identified above differ only in their position in the system.

14.2.3. Event Breakers

Event Breaker Rulesets are ordered collections of event-breaking rules that help you define the boundaries and structure of raw log data. Once you have converted unstructured raw data into structured events, you can then send them through Pipelines for further routing and data processing.

An Event Breaker Ruleset is a reusable knowledge object stored in the Knowledge Library. It acts as a parsing policy that bundles one or more individual Event Breaker rules into a single configuration that can be applied to Sources or Pipelines using the Event Breaker Function.

Each Ruleset contains a series of event breaking rules. Cribl compares incoming raw data against the filter condition for each rule in turn, based on the order that the rules appear in the Ruleset. For the first rule that matches the raw data, Cribl applies the event-breaking logic defined for that rule.



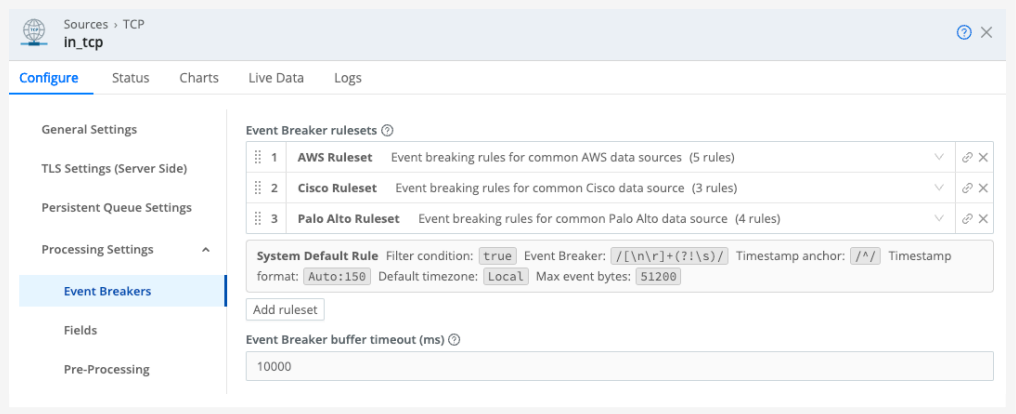
Rules within an example (AWS) ruleset that ships with Cribl Edge

Cribl provides several standard Event Breaker Rulesets for commonly used Sources out-of-the-box:

- Azure Blob Storage
- AWS
- Apache
- Bro
- Cisco
- Cribl
- Cribl Search
- CSV
- Office 365
- Palo Alto
- Splunk Search

- Wiz

You can adjust these standard Rulesets or you can create your own custom Rulesets as needed. Event Breaker rulesets shipped by Cribl will be listed under the **Cribl** tag, while user-built rulesets will be listed under **Custom**. In the case of an ID or Name conflict, the Custom pattern takes priority in listings and search.



Three Event Breaker rulesets added to a Source

How Event Breakers Work

Event Breakers consist of rulesets that can be applied to raw data one of two ways:

- **Directly on the Source:** For Event Breaker support for specific Sources, consult the documentation for that Source.
- **Using the Event Breaker Function:** You can apply event breaking directly in a Pipeline using the Event Breaker Function.

If a Source supports event breaking, you should apply Event Breaking Rulesets there. For Sources that do not support event breaking, use the the Event Breaker Function instead.

When a stream of raw data arrives at a Source or Event Breaker Function, it is processed through the Event Breaker Ruleset in a specific order of operations:

1. **Filter Condition:** When data enters the system, the engine checks it against each filter expression in sequence. If the expression evaluates to `true`, the rule configurations engage for the entire duration of that stream. Otherwise, the engine evaluates the next rule down the line. Different Source types define data streams differently:
    - **TCP Sources:** A stream equates to a single incoming TCP connection. The stream begins when the socket connection is established and ends when that connection closes.
    - **HTTP Sources:** A stream equates to a single HTTP request, with connections potentially being reused across different requests instead of corresponding to the entire connection lifecycle.

- **File-based Sources:** For Sources like File Monitor or Journal Files, each ingested file is treated as a distinct and complete stream, evaluated independently against filters.
- **Sources that support multiplexed data:** For Sources like Splunk S2S protocol, Cribl Edge demultiplexes the incoming stream based on channels. Each channel (which typically maps to a unique `source:sourcetype:host` combination in Splunk Sources) retains the same Event Breaker configuration applied consistently across its entire processing lifecycle.



Remember that Event Breaker Rulesets are evaluated in order, and the first matching rule is applied. If you define a rule with a broad or overly permissive filter, it may capture events that were intended for a more specific rule lower in the list. This means a broadly filtered rule placed high in the list can capture and incorrectly process events that were intended for a more specific rule below it.

To avoid this risk, order your rules from specific to general.

2. **The System Default Rule:** The System Default Rule is a built-in, uneditable set of Event Breaker rules that resides at the end of the ruleset order of operations. It is not displayed in the Event Breakers interface, but it automatically goes into effect for any data event that does not match the filter condition of any rule in your custom Event Breaker Ruleset. Its primary purpose is to ensure that streams of unstructured logs are broken into structured data:

Setting	Value	Explanation
Filter Condition	`true`	A condition of `true` guarantees that every event that makes it past your custom rules will be processed by the default rule.
Event Breaker	`[\n\r]+(?!\s)`	Breaks the event at a newline or carriage return, unless the following character is a whitespace (`\s`). This pattern can handle simple, single-line log formats while attempting to avoid breaking multiline events (like stack traces) where the subsequent lines are indented.
Timestamp Anchor	`^`	Starts scanning for a timestamp at the beginning of the event (`^`).
Timestamp Format	`Auto`	Uses the Auto Timestamp algorithm to automatically detect the timestamp format. This allows the default rule to handle the widest variety of logs without a pre-defined format.
Scan Depth	150 bytes	This limits how far into the event (150 bytes) the Auto Timestamp will look. It optimizes performance by reducing time scanning massive logs for a timestamp that should be near the front.
Max Event Bytes	`51200`	This sets the maximum size (50 KB) that an un-broken event can reach before Cribl forces a break, which handles events that have no newline or timestamp. It prevents a single event from consuming excessive memory.

Setting	Value	Explanation
Default Timezone	`Local`	If the timestamp does not contain timezone information, Cribl defaults to using the Worker Process's local timezone when assigning the `_time` field.

Limitations

Event Breaker Rulesets can be applied to raw data either directly on the Source or using the Event Breaker Function in Pipelines. These two methods have some limitations, as explained here.

Event Breaking in Sources

Event Breakers are directly accessible only on Sources that require incoming events to be broken into a better-defined format. For Event Breaker support for specific Sources, consult the documentation for that Source.

Always validate event breaking before turning on a Source.

Event Breaker Function

The Event Breaker Function has these limitations:

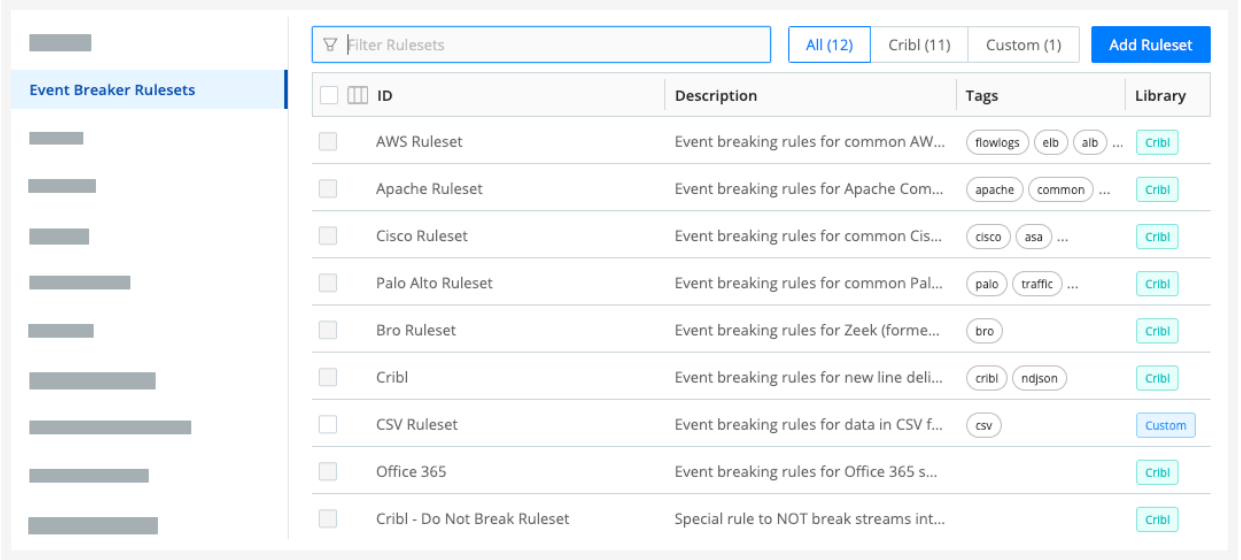
- The Event Breaker Function operates only on data in `_raw`. For other events, move the array to `_raw` and stringify it before applying the Function.
- The largest event that this Function can break is about about 128 MB (`134217728` bytes). Events exceeding this maximum size will be split into separate events, but left unbroken. Cribl Edge will set the `__isBroken` internal field to `false` on these events.

Configure Event Breaker Rulesets

You can add, edit, or customize Event Breaker Rulesets directly in Sources or Packs. You can also configure them in the Knowledge Library.

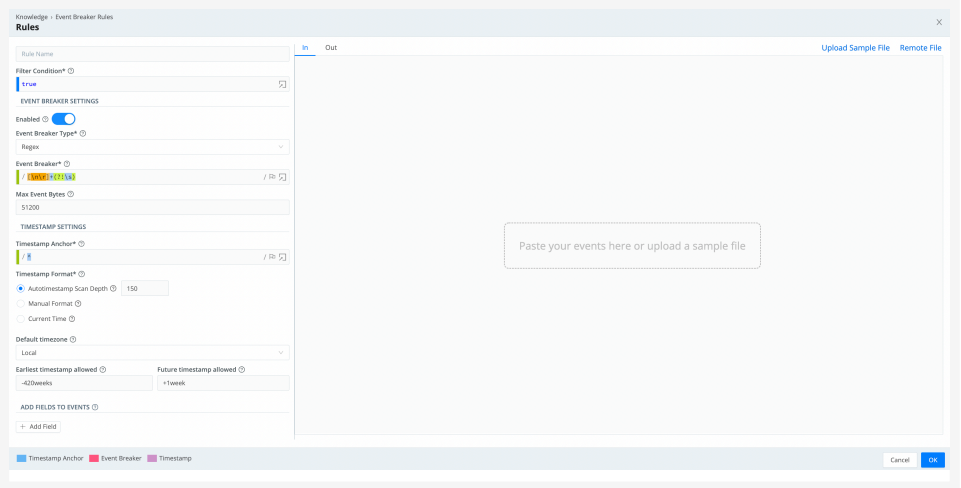
To access the Event Breakers Rulesets from the Knowledge Library:

1. Navigate to **Products** > **Cribl Edge** > **Fleets**.
2. Select a Fleet, then go to **More** > **Knowledge** > **Event Breaker Rulesets**.
3. On the resulting **Event Breaker Rulesets** page, you can edit, add, delete, search, and tag Event Breaker rules and rulesets, as necessary.



Event Breaker Rulesets page

4. Select **Add Ruleset** to create a new Event Breaker Ruleset. Enter the Rulset name and an optional description. Then select **Add Rule** within a ruleset to add a new Event Breaker.



Adding a new Event Breaker rule

5. In the **Rules** modal, configure the following settings:
    - **Name:** Enter a unique name for the rule.
    - **Filter condition:** When data enters the system, the engine checks it against each filter expression in sequence. If the expression evaluates to `true`, the rule configurations engage for the entire duration of that stream. See How Event Breakers Work for more detailed information about how this setting works.
6. In the **Event Breaker Settings** section, configure the following settings:
    - **Enabled:** By default this is toggled on. Toggle off to turn this rule off and stop it from being applied for event breaking.
    - **Event Breaker type:** Select one of the available Event Breaker types from this menu. After you select a breaker pattern, it will apply on the data stream continuously. See Event Breaker Types for

guidance on selecting an appropriate Event Breaker type. Specific Event Breakers might have additional settings, so consult the documentation for your selected type for more information.

- **Max Event Bytes:** A break will automatically occur if the accumulated event size reaches this configured byte limit, regardless of whether the regex pattern has been matched.



The highest **Max Event Bytes** value that you can set is about 128 MB (`134217728` bytes). Events exceeding that size will be split into separate events, but left unbroken. Cribl Edge will set these events' `__isBroken` internal field to `false`.

7. In the **Timestamp Settings** section, configure event timestamp settings. See Timestamp Settings for details about each setting.
8. In the **Add Fields to Events** section, you can add one or more fields to events as key-value pairs. In each field's **Value Expression**, you can fully evaluate the field value using JavaScript expressions.



Event Breakers **always** add the `cribl_breaker` field to output events. Its value is the name of the chosen ruleset. Some examples on this page omit the `cribl_breaker` field for brevity, but in real life the field is always added.

Once the rule is configured, validate it using the Data Preview window in Pipelines to see how incoming events are structured after breaking.

Timestamp Settings

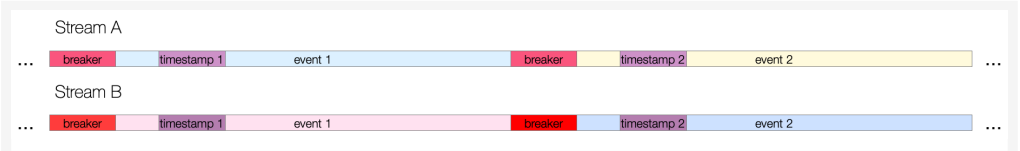
Cribl Edge timestamps events after they are synthesized from streams, ensuring accurate time representation for event processing and analysis.

- If an Event Breaker has set an event's `_time` to the current time - rather than extracting the value from the event itself - it will mark this by adding the internal field `__timestampExtracted: false` to the event.
- Cribl Edge truncates timestamps to three-digit (milliseconds) resolution, omitting trailing zeros.

Anchor: Cribl Edge first identifies a timestamp anchor within the event. This anchor serves as the starting point for timestamp extraction. The end of the first regex match is the start of the timestamp.



The closer an anchor is to the timestamp pattern, the better the performance and accuracy - especially if multiple timestamps exist within an event.



Anchors preceding timestamps

Format: Starting from the anchor, the engine attempts the following methods, stopping after a successful timestamp extraction:

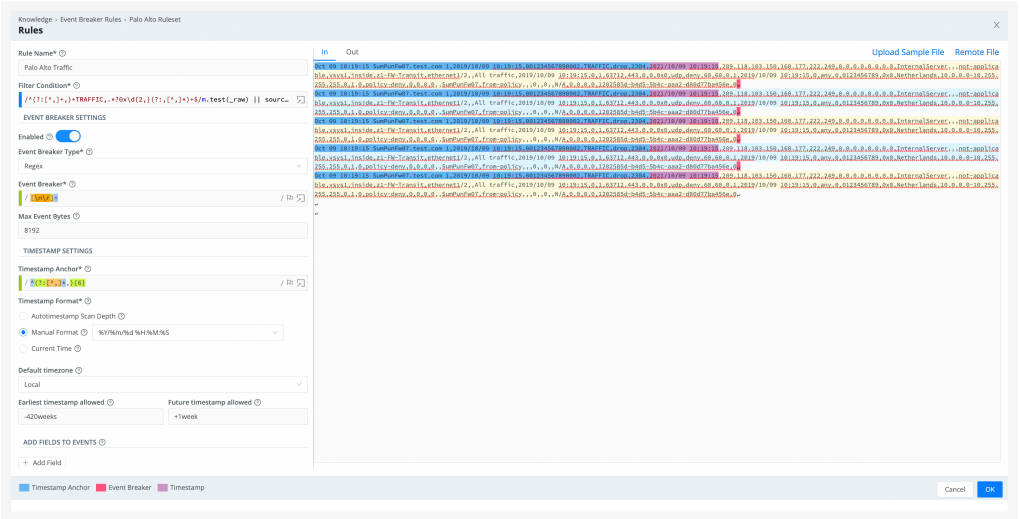
- **Autotimestamp**: Scans up to a configurable depth within the event, attempting to automatically identify and parse a timestamp. This uses the same algorithm as the Auto Timestamp Function and the `C.Time.timestampFinder()` native method.
- **Manual**: Select a `strptime` format to parse the timestamp immediately following the anchor. The anchor must lead the engine **right before** the timestamp pattern begins.
- **Current time**: The event is timestamped with the current time.

Default timezone: Assigns a timezone to timestamps that lack timezone information.

Earliest timestamp allowed and **Future timestamp allowed**: Constrain the parsed timestamp to prevent unrealistic values, setting timestamps outside these bounds to the default time.

Example Rule

This example rule breaks on newlines and uses Manual timestamping after the sixth comma, as indicated by this pattern: `^(?:[^,]*,){6}`.



An Event Breaker rule

Export and Import Rulesets

You can export and import Event Breaker Rulesets as JSON files. This facilitates sharing Rulesets among Fleets or Cribl Edge deployments.

To export a Ruleset:

1. Select to open an existing ruleset, or create a new ruleset.
2. In the resulting modal, select **Advanced Mode** to open the JSON editor.
3. Select **Export**, select a destination path, and name the file.

To import any Ruleset that has been exported as a valid JSON file:

1. Create a new ruleset.
2. In the resulting modal, select **Advanced Mode** to open the JSON editor.
3. Select **Import**, and choose the file you want.
4. Select **OK** to get back to the **New Ruleset** modal.
5. Select **Save**.



Every Ruleset must have a unique value in its top `id` key. Importing a JSON file with a duplicate `id` value will fail at the final **Save** step, with a message that the ruleset already exists. You can remedy this by giving the file a unique `id` value.

Build Event Breakers From Files (Edge Only)

You can use Cribl Edge to apply and author Event Breakers while exploring files, including files that you have not saved as sample files. This option includes remote files that are viewable only from Edge. See Exploring Cribl Edge on Linux or Exploring Cribl Edge on Windows.

Troubleshoot Event Breakers

If you notice fragmented events, check whether Cribl Edge has added a `__timeoutFlush` internal field to them. This diagnostic field's presence indicates that the events were flushed because the Event Breaker buffer timed out while processing them. These timeouts can be due to large incoming events, backpressure, or other causes.

Specify Minimum `_raw` Length

If you notice that the same Collector or Source is applying inconsistent Event Breakers to different datasets - such as intermittently falling back to the **System Default Rule** - you can address this by adjusting Breakers' **Min Raw Length** setting.

When determining which Event Breaker to use, Cribl Edge creates a substring from the incoming `_raw` field, at the length set by rules' **Max Event Bytes** parameters. If Cribl Edge does not get a match, it checks whether the substring of `_raw` has at least the threshold number of characters set by **Min Raw Length**. If so, Cribl Edge does not wait for more data.

You can tune this **Min Raw Length** threshold to account for varying numbers of inapplicable characters at the beginning of events. The default is `256` characters, but you can set this as low as `50`, or as high as `100000` (100 KB).

14.2.4. Event Breaker Types

You can apply several types of Event Breakers to incoming data streams (listed alphabetically):

- Azure Virtual Network (VNet) Flow: For Azure VNet flow logs from Azure Blob Storage and Azure Event Hubs.
- CSV: For data that adheres to the CSV standard, such as output from databases, financial logs, spreadsheets, or network device reports. This Event Breaker handles row delineation, field extraction, and quoting rules for CSV data.
- File Header: For logs that use a standard file header structure, such as Bro, IIS, or Apache access logs. Use the File Header Breaker instead of Regex when the field names are complex but consistent across the log file.
- JSON Array: For single, large JSON objects containing a nested array of records. This Event Breaker is particularly useful for logs from services where a single log entry contains an array of event records, such as AWS CloudTrail, EKS CloudWatch, or Google Cloud Audit Logs.
- JSON Newline Delimited: For log data where each event is a complete, self-contained JSON object that is followed by a newline. This format is widely used by cloud providers, modern logging agents, and data platforms.
- Regex: For any log data that doesn't fit the structured formats of the CSV, JSON Array, or File Header breakers. This includes multi-line logs where a single event spans multiple lines and non-standard single-line logs that use custom formats.
- Timestamp: For log streams that use non-standard or highly varied timestamp formats, especially where writing a single, complex regex would be difficult or inefficient. This Event Breaker is also good for multi-line logs where every new event begins with a timestamp, but the exact format isn't guaranteed or known ahead of time.

If you're unsure which Event Breaker to use, the Regex Event Breaker is a good default Event Breaker because of its flexibility.

See Event Breakers for general information about event breakers.

14.2.4.1. Azure Virtual Network (VNet) Flow Event Breaker

The Azure Virtual Network (VNet) Flow Event Breaker segments and structures the highly nested JSON data generated by Azure VNet logs. It takes in raw data that uses the Azure VNet flow logs schema to identify individual flow tuples, then outputs the data into a structured data format.

Use this Event Breaker to ingest Azure VNet flow logs from Azure Blob Storage.

See Event Breakers for general information about event breakers.



Azure has deprecated Network Security Group (NSG) flow logs in favor of Azure VNet Flow logs. To optimize system performance, use this Azure VNet Flow Event Breaker to de-aggregate records at the Source. This is more efficient than ingesting raw blobs and using the JSON Unroll Function later in your Pipeline.

Settings

Selecting the Azure VNet Flow Event Breaker type does not expose any additional settings beyond the standard Event Breaker settings. See Event Breakers for a description of the available settings.

Configuration Example

The following is an example of data input before the Azure VNet Flow Event Breaker processes it:

Example raw input

```
{
    "records": [
        {
            "time": "2022-09-14T09:00:52.5625085Z",
            "flowLogVersion": 4,
            "flowLogGUID": "66aa66aa-bb77-cc88-dd99-00ee00ee00ee",
            "macAddress": "112233445566",
            "category": "FlowLogFlowEvent",
            "flowLogResourceID": "/SUBSCRIPTIONS/aaaa0a0a-bb1b-cc2c-dd3d-eeeeee4e4e4e
            "targetResourceID": "/subscriptions/aaaa0a0a-bb1b-cc2c-dd3d-eeeeee4e4e4e/
            "operationName": "FlowLogFlowEvent",
            "flowRecords": {
                "flows": [
                    {
                        "aclID": "00aa00aa-bb11-cc22-dd33-44ee44ee44ee",
                        "flowGroups": [
                            {
                                "rule": "DefaultRule_AllowInternetOutBound",
                                "flowTuples": [
                                    "1663146003599,10.0.0.6,192.0.2.180,23956,443,6,O
                                    "1663146003606,10.0.0.6,192.0.2.180,23956,443,6,O
                                    "1663146003637,10.0.0.6,203.0.113.17,22730,443,6,
                                    "1663146003640,10.0.0.6,203.0.113.17,22730,443,6,
                                    "1663146004251,10.0.0.6,203.0.113.17,22732,443,6,
                                    "1663146004251,10.0.0.6,203.0.113.17,22732,443,6,
                                    "1663146004622,10.0.0.6,203.0.113.17,22734,443,6,
                                    "1663146004622,10.0.0.6,203.0.113.17,22734,443,6,
                                    "1663146017343,10.0.0.6,198.51.100.84,36776,443,O
                                    "1663146022793,10.0.0.6,198.51.100.84,36776,443,O
                                ]
                            }
                        ]
                    },
                    {
                        "aclID": "00aa00aa-bb11-cc22-dd33-44ee44ee44ee",
                        "flowGroups": [
                            {
                                "rule": "BlockHighRiskTCPPortsFromInternet",
                                "flowTuples": [
                                    "1663145998065,101.33.218.153,10.0.0.6,55188,22,O
                                    "1663146005503,192.241.200.164,10.0.0.6,35276,119
                                ]
                            `
```

```
                        },
                        {
                            "rule": "Internet",
                            "flowTuples": [
                                "1663145989563,192.0.2.10,10.0.0.6,50557,44357,6,
                                "1663145989679,203.0.113.81,10.0.0.6,62797,35945,
                                "1663145989709,203.0.113.5,10.0.0.6,51961,65515,6
                                "1663145990049,198.51.100.51,10.0.0.6,40497,40129
                                "1663145990145,203.0.113.81,10.0.0.6,62797,30472,
                                "1663145990175,203.0.113.5,10.0.0.6,51961,28184,6
                                "1663146015545,192.0.2.10,10.0.0.6,50557,31244,6,
                            ]
                        }
                    ]
                }
            ]
        }
    ]
}
```

Copy

Output

The Azure VNet Flow Event Breaker unrolls the nested `flowTuples` array and maps the comma-separated values into structured fields. It flattens the metadata (like the `rule` and `resourceId`) into each output event, ensuring that context is preserved even after the original large object is split.

From the example raw data, the Azure VNet Event Breaker would generate two output events:

Example Output

```
{
  "_raw": "1663146003599,10.0.0.6,192.0.2.180,23956,443,6,O,B,NX,0,0,0,0",
  "flowLogVersion": 4,
  "flowLogGUID": "66aa66aa-bb77-cc88-dd99-00ee00ee00ee",
  "macAddress": "112233445566",
  "category": "FlowLogFlowEvent",
  "flowLogResourceID": "/SUBSCRIPTIONS/aaaa0a0a-bb1b-cc2c-dd3d-eeeeee4e4e4e/RESOURCEC
  "targetResourceID": "/subscriptions/aaaa0a0a-bb1b-cc2c-dd3d-eeeeee4e4e4e/resourceGr
```

14.2.4.2. CSV Event Breaker

The CSV Event Breaker can take in Comma Separated Value (CSV) data streams and automatically extract fields. It can process structured, tabular data where a header row defines the field names for all subsequent rows.

Use this Event Breaker:

- Whenever you are ingesting data that adheres to the CSV standard.
- For output from databases, financial logs, spreadsheets, or network device reports. The Event Breaker handles row delineation, field extraction, and quoting rules for this type of data.

See Event Breakers for general information about event breakers.

Settings

Selecting the CSV Event Breaker type exposes these additional settings:

- **Delimiter**: The character used to separate or split individual field values within a row. For standard CSV, the default is a comma (`,`). Change the delimiter for TSV (Tab Separated Values) or when dealing with logs that use a pipe (`|`) or semicolon (`;`) as a separator. Defaults to `,`.
- **Quote Char**: The character used to enclose and protect field values that contain the delimiter character itself. For example, if a field is `"10,000"`, the quotes tell the breaker that the comma is part of the value, not a separator. Defaults to `"`.
- **Escape Char**: The character used to indicate that the next character should be treated literally. It is often used to escape the **Quote Char** itself within a quoted field. For example, to include a literal double-quote inside a quoted field, you might use an escape character: `"He said, ""Hello.""` Defaults to `"`.

Configuration Example

The following is an example of data input before the CSV Event Breaker processes it:

Example raw input - CSV format

```
time,host,source,model,serial,bytes_in,bytes_out,cpu
1611768713,"myHost1","anet","cisco","ASN4204269",11430,43322,0.78
1611768714,"myHost2","anet","cisco","ASN420423",345062,143433,0.28
```

Copy

Output

The CSV Event Breaker treats the first line of the input as the header row and automatically extracts those names as event fields for all subsequent rows. It will also properly add quotes around all values in the `_raw` field of the output events, regardless of their initial state.

From the example raw data, the CSV Event Breaker would generate two output events:

Example CSV Output as JSON


```
{
  "_raw": "\"1611768713\",\"myHost1\",\"anet\",\"cisco\",\"ASN4204269\",\"11430\",\"4
  "time": "1611768713",
  "host": "myHost1",
  "source": "anet",
  "model": "cisco",
  "serial": "ASN4204269",
  "bytes_in": "11430",
  "bytes_out": "43322",
  "cpu": "0.78",
  "_time": 1611768713
}
{
  "_raw": "\"1611768714\",\"myHost2\",\"anet\",\"cisco\",\"ASN420423\",\"345062\",\"1
  "time": "1611768714",
  "host": "myHost2",
  "source": "anet",
  "model": "cisco",
  "serial": "ASN420423",
  "bytes_in": "345062",
  "bytes_out": "143433",
  "cpu": "0.28",
  "_time": 1611768714
}
```


Copy

14.2.4.3. File Header Event Breaker

The File Header Event Breaker ingests highly structured log formats where the field names are explicitly defined in a header section at the top of the file. This Event Breaker relies on a combination of regular expressions to first identify the header, and then use the captured field names to parse all subsequent data lines.

Use this Event Breaker:

- For logs that use a standard file header structure, such as Bro, IIS, or Apache access logs.
- Instead of a simple Regex Breaker when the field names are complex but consistent across the log file.

See Event Breakers for general information about event breakers.

Settings

The File Header Event Breaker requires you to define the structure of the log using a set of coordinated regular expressions:

- **Header line:** The regex used to match and identify a line as part of the header section. Any line that matches this pattern will be treated as metadata and not as an event. For example, this expression matches any line starting with a hash symbol: `^#`.
- **Field delimiter:** The regex used to split the field names captured by the Field Regex into individual field names. This delimiter is also implicitly used to break the subsequent data lines into values. For example, this expression matches one or more whitespace characters, common in space-separated log formats (such as Bro): `\s+`.
- **Fields regex:** This is the most critical regex. It must have exactly one capturing group that isolates the list of field names from the rest of the header text. This captured group is what the Field Delimiter will then process. For example, this expression matches a line starting with `#fields` and captures everything that follows the label into a single group: `^#[Ff]ields[:]?\s+(.*)`.
- **Null value:** The string representation of a null or empty value in the data lines. Fields with this value will not be added to the event object, keeping the event clean. The field is blank by default. Enter a string such as `-` if necessary, which is common in firewall or security logs.
- **Clean fields:** Toggling this setting on will sanitize the extracted field names by replacing any non-alphanumeric characters with an underscore (`_`). This is highly recommended to ensure compatibility with downstream systems like Splunk or Elasticsearch. For example, a field named `id.orig_h` becomes `id_orig_h`.

Configuration Example

The following is an example of data input before the File Header Event Breaker processes it. The header section is typically at the top of the file. It can be a single line or more:

Example raw input - File Header format

```
#fields ts      uid     id.orig_h       id.orig_p       id.resp_h       id.resp_p
#types  time    string  addr    port    addr    port    enum
1331904608.080000       -     192.168.204.59  137     192.168.204.255 137     udp
1331904609.190000       -     192.168.202.83  48516   192.168.207.4   53      udp
```

Copy

Example Settings

Setting	Value	Purpose
Header Line	`^#`	Matches all header lines.
Field Delimiter	`\s+`	Separates fields by whitespace.
Field Regex	`^#[Ff]ields[:]?\s+(.*)`	Captures the field names.
Clean Fields	On	Converts `id.orig_h` to `id_orig_h`.

Output

In the output, the Header Line (starting with `#fields`) and the Type Line (starting with `#types`) are ignored. The field names extracted from the Field Regex line are used to key the values in the output events.

From the example raw data, the File Header Event Breaker would generate two output events:

Example Output as JSON

```
{
  "_raw": "1331904608.080000       -     192.168.204.59  137     192.168.204.255 137
  "ts": "1331904608.080000",
  "id_orig_h": "192.168.204.59",
  "id_orig_p": "137",
  "id_resp_h": "192.168.204.255",
  "id_resp_p": "137",
  "proto": "udp",
  "_time": 1331904608.08
}
{
  "_raw": "1331904609.190000       -     192.168.202.83  48516   192.168.207.4   53
  "ts": "1331904609.190000",
  "id_orig_h": "192.168.202.83",
  "id_orig_p": "48516",
  "id_resp_h": "192.168.207.4",
  "id_resp_p": "53",
  "proto": "udp",
```

14.2.4.4. JSON Array Event Breaker

The JSON Array Event Breaker handles log formats where multiple distinct records are bundled together within a single top-level JSON object. It takes one large event and performs a targeted unrolling operation: it identifies an array nested within that parent JSON object and splits each element of that array into a separate, individual event.

Use this Event Breaker for:

- Logs where a single log entry contains an array of event records.
- Sources like AWS CloudTrail, EKS CloudWatch, or Google Cloud Audit Logs.



If a Source supports Event Breakers (such as AWS Sources), it is much more efficient to unroll JSON in an Event Breaker rather than a Pipeline Function.

See Event Breakers for general information about event breakers.

Settings

The JSON Array Event Breaker has these settings:

- **Array field:** Optional path to array in a JSON event with records to extract. For example, `Records`.
- **Parent fields to copy:** This setting allows you to retain key metadata from the original parent event and include it with each of the extracted child events. By default, only the extracted events from the array are passed down the Pipeline. For example, you can use this to copy `host`, `source`, `region`, or other contextual information that is not contained within the array itself.

  To specify which fields to copy, enter a comma-separated list of top-level field names. You can also use `*` as a wildcard to copy all top-level fields. Be aware that:

  - This setting only supports copying fields from the top level of the JSON document. Nested fields are not supported.
  - The `Array field` you specify to be unrolled is always excluded from the copied fields. This prevents redundant data.
  - Enclose field names containing special characters (such as `my-field`) in single or double quotes.
- **JSON extract fields:** This toggle controls whether Cribl Edge parses the JSON array and automatically extracts key-value pairs from each event:

- When enabled, Cribl Edge will fully parse each JSON object and extract all fields (such as `user_id`, `status`, `event_type`). This is the ideal setting for sending data to Destinations that need structured data for analysis like Splunk or Elasticsearch for searching, filtering, and dashboard-building. When this toggle is on, it exposes the **Timestamp field** setting.
- When disabled, Cribl Edge will not extract any fields from the JSON objects. Only the `_raw` (the full event text) and `time` will be available. Choose this option if your primary goal is to send raw data for archival purposes with the highest possible throughput.



The **JSON extract fields** toggle impacts the **Timestamp format** setting in the Timestamp Settings. If the **JSON extract fields** toggle is off, do not select the **Manual format** option. If the toggle is on, you can select any option.

- **Timestamp field:** This optional setting appears when **JSON extract fields** is toggled on. This allows you to specify a path to the timestamp field within each extracted event. For example, `eventTime` or `level1.level2.eventTime`. Cribl Edge will use this value to set the event's `_time` field.
- **Max event bytes**: The maximum size (in number of bytes) that an event can be before being flushed to the Pipelines.
- **Event byte limit**: The maximum size (in number of bytes) that an event can be before being flushed to the Pipelines.

Configuration Examples

The following is an example of data input before the JSON Array Event Breaker processes it:

Example raw input - JSON (Array)

```
{
  "someField": 42,
  "someArray": [1, 2, 3],
  "myObjectArray": [
    {"one": 1, "two": 2},
    {"three": 3, "four": 4}
  ],
  "Records": [
    { "eventVersion": "1.0" },
    { "eventVersion": "2.0" }
  ],
  "myObject": {
    "someField": {
      "test1": 1,
      "test2": "two"
    },
    anotherArray: [
      {"id": 1},
      {"id": 2}
    ]
  }
}
```

Copy

Example 1 - Copy All Top-Level Fields

This first example shows how to use the wildcard `*` to copy all top-level fields from the original JSON event to each of the new, extracted events.

JSON Configuration | User Interface Configuration

```
{
  "parentFieldsToCopy": ["*"],
  "jsonArrayField": "Records"
}
```

Copy

With these configurations, the original raw data will have this output:

Example 1 - Copy All Top-Level Fields

```
[
  {
    _raw: "{ "eventVersion": "1.0" }",
    "someField": 42,
    "someArray": [1, 2, 3],
    "myObjectArray": [
      {"one": 1, "two": 2},
      {"three": 3, "four": 4}
    ],
    "myObject": {
      "someField": {
        "test1": 1,
        "test2": "two"
      },
      anotherArray: [
        {"id": 1},
        {"id": 2}
      ]
    }
  },
  {
    _raw: "{ "eventVersion": "2.0" }",
    "someField": 42,
    "someArray": [1, 2, 3],
    "myObjectArray": [
      {"one": 1, "two": 2},
      {"three": 3, "four": 4}
    ],
    "myObject": {
      "someField": {
        "test1": 1,
        "test2": "two"
      },
      anotherArray: [
        {"id": 1},
        {"id": 2}
      ]
    }
  }
]
```

Copy

Example 2 - Copy Specific Fields

This example shows how to select only specific top-level fields to copy. By providing a comma-separated list of field names, you can control which metadata carries over to the extracted events, resulting in smaller events and reduced data volume.

JSON Configuration | User Interface Configuration

```
{
  "parentFieldsToCopy": ["someField", "myObject"],
  "jsonArrayField": "Records"
}
```

Copy

With these configurations, the original raw data will have this output:

Example 2 - Copy Specific Fields

```
[
  {
    _raw: "{ "eventVersion": "1.0" }",
    "someField": 42,
    "myObject": {
      "someField": {
        "test1": 1,
        "test2": "two"
      },
      anotherArray: [
        {"id": 1},
        {"id": 2}
      ]
    }
  },
  {
    _raw: "{ "eventVersion": "2.0" }",
    "someField": 42,
    "myObject": {
      "someField": {
        "test1": 1,
        "test2": "two"
      },
      anotherArray: [
        {"id": 1},
        {"id": 2}
      ]
    }
  }
]
```

Copy

Example 3 - Use a Wildcard with a Prefix

You can also use a wildcard with a prefix to copy fields that share a common name. This example uses `some*` to copy any top-level fields that start with "some," such as `someField` and `someArray`. This is useful when you have a consistent naming convention for related fields.

JSON Configuration | User Interface Configuration

```
{
  "parentFieldsToCopy": ["some*"],
  "jsonArrayField": "Records"
}
```

Copy

14.2.4.5. JSON New Line Delimited Event Breaker

The JSON New Line Delimited Event Breaker handles Newline Delimited JSON (NDJSON) data streams. It treats every newline character (`\n`) as an event boundary, and then automatically parses the JSON object on that line, extracting all key-value pairs as fields. This format is widely used by cloud providers, modern logging agents, and data platforms.

Use this Event Breaker:

- For log data where each event is a complete, self-contained JSON object that is followed by a newline.



Because this Event Breaker relies solely on the newline character, you cannot use it to parse multi-line JSON objects, or single events that contain embedded newlines. Instead, use either the JSON Array Event Breaker or Regex Event Breaker.

See Event Breakers for general information about event breakers.

Settings

This Event Breaker does not require any additional settings beyond the basic Event Breaker settings. That is because it uses a strict format:

- Events are automatically broken by a newline character (`\n`).
- All top-level fields within each JSON object are automatically extracted and added to the event.

Configuration Example

The following is an example of data input before the JSON New Line Delimited Event Breaker processes it:

Example raw input - NDJSON

```
{"time":"2020-05-25T18:00:54.201Z","cid":"w1","channel":"clustercomm","level":"info",
{"time":"2020-05-25T18:00:54.246Z","cid":"w0","channel":"clustercomm","level":"info",
```

Copy

Output

The JSON New Line Delimited Event Breaker processes an NDJSON stream by treating each line of the input as a separate event. All fields are automatically extracted. The event breaks cleanly at the end of each JSON object.

From the example raw data, the JSON New Line Delimited Event Breaker would generate two output events:

Example CSV Output as JSON


```
{
  "_raw": "{\"time\":\"2020-05-25T18:00:54.201Z\",\"cid\":\"w1\",\"channel\":\"cluste
  "time": "2020-05-25T18:00:54.201Z",
  "cid": "w1",
  "channel": "clustercomm",
  "level": "info",
  "message": "metric sender",
  "total": 720,
  "dropped": 0,
  "_time": 1590429654.201,
}
{
  "_raw": "{\"time\":\"2020-05-25T18:00:54.246Z\",\"cid\":\"w0\",\"channel\":\"cluste
  "time": "2020-05-25T18:00:54.246Z",
  "cid": "w0",
  "channel": "clustercomm",
  "level": "info",
  "message": "metric sender",
  "total": 720,
  "dropped": 0,
  "_time": 1590429654.246,
}
```


Copy

14.2.4.6. Regex Event Breaker

The Regex Event Breaker is the default and most flexible Event Breaker type, allowing you to define event boundaries using regular expressions (regex). It gives you more granular control over how unstructured or non-standard logs are segmented into individual events.



Event Breaker patterns use the JavaScript (ECMAScript) regex engine. If you test patterns in external tools such as Regex101, select the **ECMAScript (JavaScript)** flavor. For more information, see Regex Flavor.

Use this Event Breaker for:

- Any log data that doesn't fit the structured formats of the CSV, JSON Array, or File Header breakers.
- Multi-line logs where a single event spans multiple lines and non-standard single-line logs that use custom formats.

See Event Breakers for general information about event breakers.

Considerations

When using the Regex Event Breaker, keep this in mind:

- **Breaks are continuous:** The pattern applies continuously to the data stream. Any content before the match is considered part of the current event, and the break occurs at the start of the match.
- **Consuming and retaining content:** By default, the matched content of the Event Breaker regex is consumed (thrown away). To define the break point without discarding the content that signifies the new event (such as a timestamp), use a positive lookahead regex, such as `(?=pattern)`. This is the standard practice for multiline logs.
- **Avoid capturing groups:** Do not use capturing groups (parentheses like `(pattern)`) inside the Event Breaker pattern. Using them will cause further, often unintended, splitting of the stream.

Settings

The Regex Event Breaker has two relevant settings:

- **Event Breaker:** The regular expression that identifies the start of a new event. Breaking occurs at the beginning of the matched pattern.

- **Max Event Bytes:** A break will automatically occur if the accumulated event size reaches this configured byte limit, regardless of whether the regex pattern has been matched.

Configuration Example

The following is an example of multi-line log data input before the Regex Event Breaker processes it:

Example raw input - Non-standard log format

```
2020-05-19 16:32:12 moen3628 ipsum[5213]: Use the mobile TCP feed, then you can progr
   Try to connect the FTP sensor, maybe it will connect the digital bus!
   Try to navigate the AGP panel, maybe it will quantify the mobile alarm!
2020-05-19 16:32:12 moen3628 ipsum[5213]: Use the mobile TCP feed, then you can progr
   Try to connect the FTP sensor, maybe it will connect the digital bus!
   Try to navigate the AGP panel, maybe it will quantify the mobile alarm!
```

Copy

Example Settings

Setting	Value	Purpose
Event Breaker	`[\n\r]+(?=\d+-\d+-\d+\s\d+:\d+:\d+)`	This setting breaks after a newline or carriage return, but only if followed by a timestamp pattern.
Max Event Bytes	`51200`	The default setting.

Output

From the example raw data, the Regex Event Breaker would generate two output events:

Example Output as JSON

```
{
  "_raw": "2020-05-19 16:32:12 moen3628 ipsum[5213]: Use the mobile TCP feed, then yo
  "_time": 1589920332
}
{
```

14.2.4.7. Timestamp Event Breaker

The Timestamp Event Breaker is a low-configuration method to break events at the start of any line where Cribl Edge detects a valid timestamp.

Use this Event Breaker for:

- Log streams that use non-standard or highly varied timestamp formats where writing a single, complex regex would be difficult or inefficient.
- Multi-line logs (like application stack traces) where every new event begins with a timestamp, but the exact format isn't guaranteed or known ahead of time.



While highly flexible, the Timestamp Breaker may have a slight performance trade-off compared to a precise Regex Breaker because it requires more processing to dynamically detect the timestamp pattern. For maximum precision and performance on known log formats, use a Regex Event Breaker with a positive lookahead instead.

See Event Breakers for general information about event breakers.

Settings

The Timestamp Event Breaker is useful as an out of the box Event Breaker for event segmentation:

- There is no field to input a regex or pattern.
- A new event is defined when Cribl Edge identifies a valid time value at the beginning of a line.

Configuration Example

The following is an example of data input before the Timestamp Event Breaker processes it:

Example raw input - Timestamp format

```
{"level":"debug","ts":"2021-02-02T10:38:46.365Z","caller":"sdk/sync.go:42","msg":"Har
{"level":"debug","ts":"2021-02-02T10:38:56.365Z","caller":"sdk/sync.go:42","msg":"Har
```

Copy

Output

From the example raw data, the Timestamp Event Breaker would generate two output events:

Example CSV Output as JSON


```
{
  "_raw": "{\"level\":\"debug\",\"ts\":\"2021-02-02T10:38:46.365Z\",\"caller\":\"sdk/
  "_time": 1612262326.365
}
{
  "_raw": "{\"level\":\"debug\",\"ts\":\"2021-02-02T10:38:56.365Z\",\"caller\":\"sdk/
  "_time": 1612262336.365
}
```


Copy

14.3. Routes

Before incoming events are transformed by a processing Pipeline, Cribl Edge uses a set of filters to first select a **subset** of events to deliver to the correct Pipeline. This selection is normally made via Routes.

> **Don't Need Routes?**
>
> Routes are designed to filter, clone, and cascade incoming data across a related set of Pipelines and Destinations. If all you need are independent connections that link parallel Source/Destination pairs, you can use Cribl Edge's QuickConnect rapid visual configuration tool as an alternative to Routes.

Accessing Routes

To access Routes:

1. In the sidebar, select **Fleets**, then select a Fleet.
2. On the **Fleets** submenu, select **More**, then **Data Routes**.

   > On a Single-instance deployment, select **More** straight from the **Home** submenu.

3. To configure a new Route, select **Add Route**.

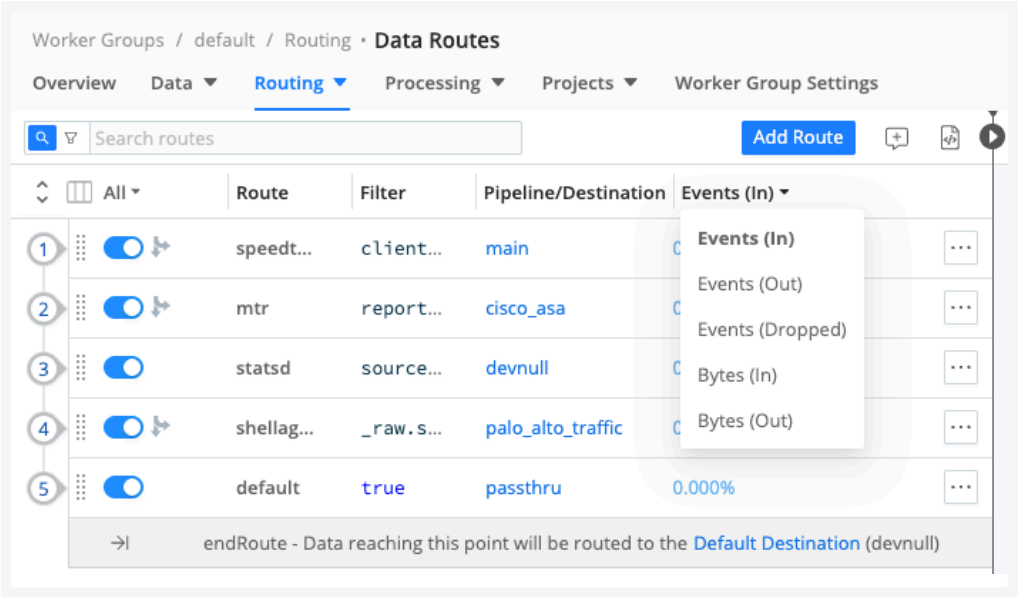
How Do Routes Work

Routes apply filter expressions on incoming events to send matching results to the appropriate Pipeline or Pack. Filters are JavaScript-syntax-compatible expressions that are configured with each Route. Examples are:

- `true`
- `source=='foo.log' && fieldA=='bar'`

> There can be multiple Routes in a Cribl Edge deployment, but each Route can be associated with only **one processing** Pipeline or Pack. (Sources and Destinations can also have their own attached pre-/post-processing Pipeline or Pack.)
>
> Each Route must have a unique name. The **Route Name** field is case-sensitive, so entering the same characters with different capitalization will create a Route with a separate name.

Routes are evaluated in their display order, from the first Route in the list to the last. The configuration in the following screenshot means that events are evaluated first against the Route named **speedtest**, then against **mtr**, then against **statsd**, and so on. At the end, the **default** Route serves as a catch-all to explicitly route events that do not match any of the Route filters. We recommend always using a catch-all Route.



Routes and bytes

Use the drop-down menu for the **Events (In)** column to select whether the column displays Events In, Out, or Dropped or Bytes In or Out. The stats shown in this column are for the most recent 15 minutes.

Displayed percentages of Events or Bytes In and Out are by comparison to throughput or filtering across all Routes. For example, if you have four Routes that share an identical filter, each Route's Events In might show approximately 25% of the total inbound traffic across all Routes.

Design data paths to move through as few Routes as possible. This usually means we want to reduce volume of events as early as possible. If most of the events being processed are sent to a Destination by the first Route, this will spare all the other Routes and Pipelines wasting processing cycles and testing against whether filter criteria are met.

Manage the Routes Page

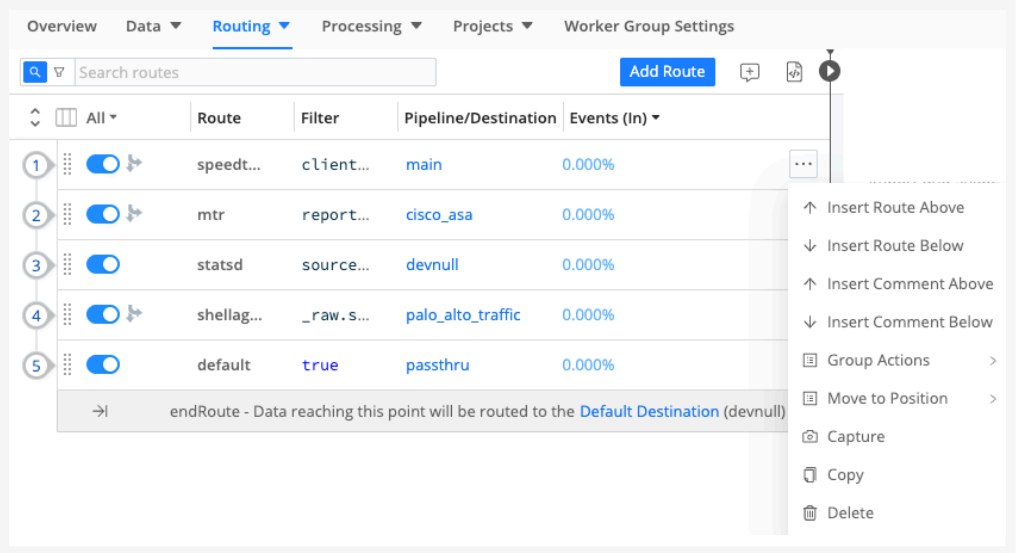
To apply a Route before another, simply drag it vertically. Toggle Routes or or off inline, as necessary, to facilitate development and debugging.

You can press the] (right-bracket) shortcut key to toggle between the Data Preview pane and the expanded Routes display shown above. (This works when no field has focus.)

Click a Route's Options (•••) menu to display multiple options:

- Insert, group, move, copy, or delete Routes.
- Insert comments above or below Routes.

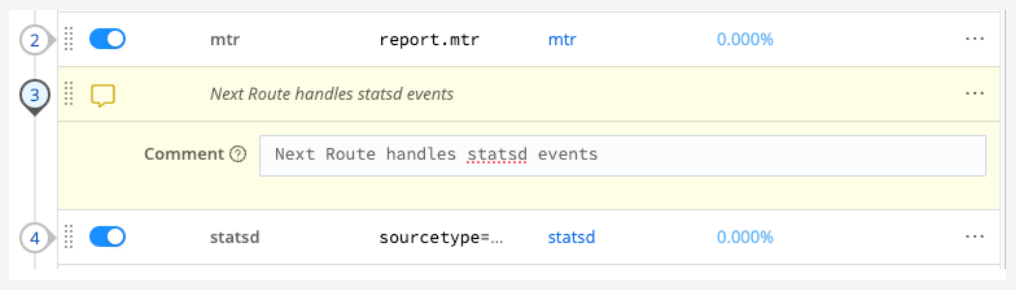
- Capture sample data through a selected Route.



Route Options menu

Copying a Route displays a confirmation message and adds a Paste button next to **Add Route**. Pasting creates an exact duplicate of the Route, with a warning indicator to change its duplicate name.

You can use comments to describe Routes' purposes and/or interactions.



Comments make the Routing table self-documenting

Destination

You can configure each Route to include a **Destination** to send events to, after they're processed by the Pipeline or Pack.

If you toggle **Enable Expression** on, the **Destination** field changes to a **Destination expression** field. Here, you can enter a JavaScript expression that Cribl Edge will evaluate as the name of the Destination. Sample expression format: `` `myDest_${C.logStreamEnv}` ``. (This evaluation happens at Route construction time, not per event.)

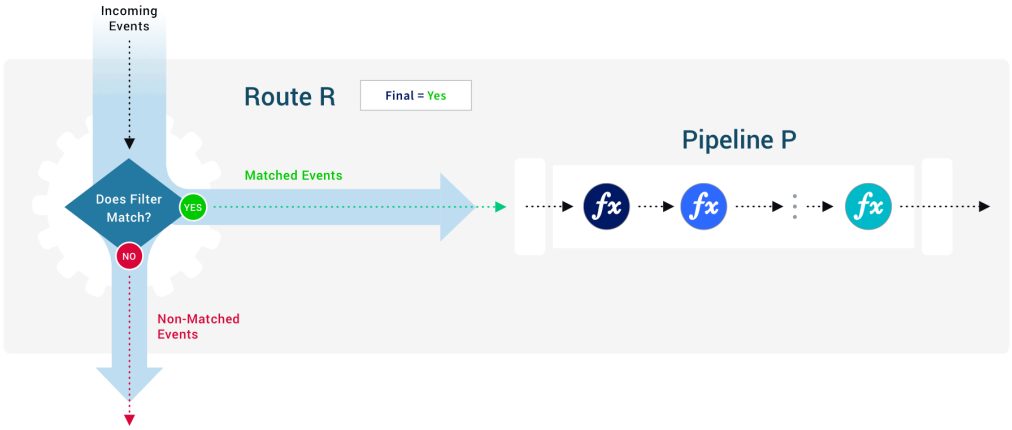


Destination expression enabled

The Final Toggle

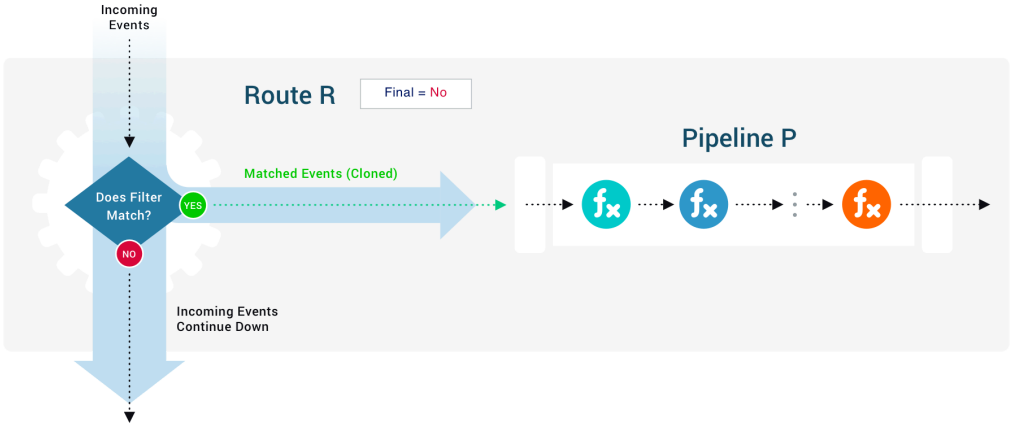
The **Final** toggle in Route settings controls what happens to an event after a Route-Pipeline pair matches it.

When **Final** is toggled on (default), the Route processes matched events and sends them to the appropriate Pipeline or Pack. Matched events **do not** continue to the next Route. Non-matched events **do** continue to the next Route.



Data flow for a Route with Final toggled on

When **Final** is toggled off, the Route still processes matched events and sends them to a Pipeline or Pack. However, all events (matched and non-matched) continue to the next route.



Data flow for a Route with Final toggled off

When you toggle **Final** off, the event sent to the current Route technically becomes a *clone* of the original event. The **Add Clone** button lets you add a fields - names and values - to the cloned events sent to this Route's Pipeline.



Non-final Route: add clone fields

Event Cloning Strategy

Depending on your cloning needs, you might want to follow a **most-specific first** or a **most-general first** processing strategy. The general goal is to minimize the number of filters/Routes an event gets evaluated against. For example:

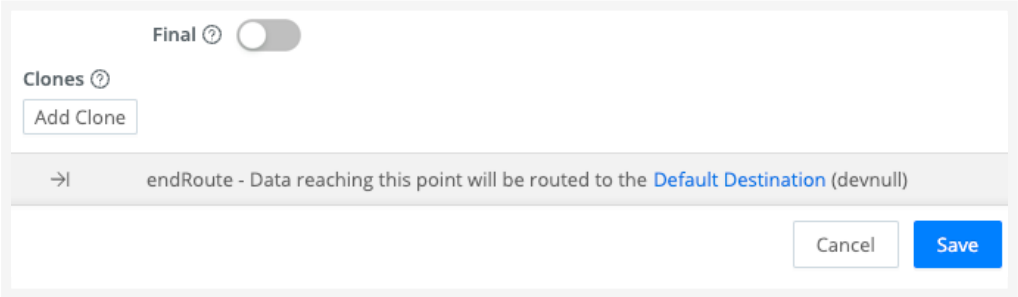
If you don't need to clone any events (that is, **Final** is toggled on everywhere), then it makes sense to start with the broadest expression at the top. This will consume as many events as early as possible.

If you need to clone a narrow set of events, then it might make sense to do that upfront. Then, follow it with a Route that consumes those clones immediately after.

Clone events as late as possible in the Routes. This will minimize the number of Functions acting on data, to the extent possible.

The `endRoute` Bumper

The `endRoute` bumper appears at the bottom of the Routing table. This is a reminder that, if **no** Route in the table has **Final** toggled on, events will continue to Cribl Edge's configured Default Destination.



endRoute warning and link

This is a backstop to ensure data flow. However, if that configured default is also configured as the **Destination** of a Route higher in the table, duplicate events will reach that Destination.

You can correct this either by toggling **Final** on for a Route or by changing the Default Destination. The bumper provides a link to the Default Destination's config, and identifies the currently configured default in parentheses.

Route Groups

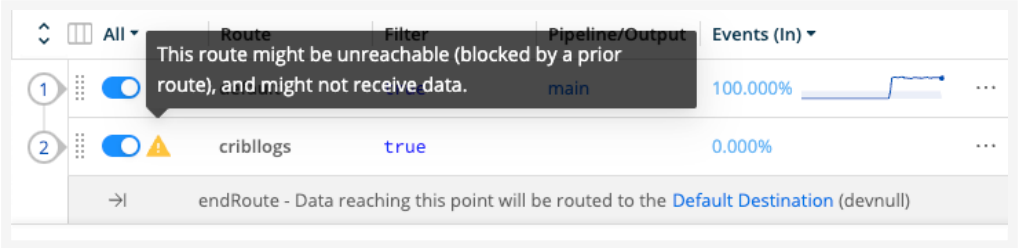
A Route Group is a collection of consecutive Routes that can be moved up and down the Route stack together. Groups help with managing long lists of Routes. They are a UI visualization only: While Routes are in a group, those Routes maintain their global position order.



Route groups work much like Function groups, offering similar UI controls and drag-and-drop options.

Unreachable Routes

Routes display an “unreachable” warning indicator (orange triangle) when data can’t reach them.



Unreachable Route warning, on hover

This condition will occur when, with your current configuration, any Route higher in the stack matches **all** three of these conditions:

- Previous Route is enabled.
- Previous Route’s **Final** toggle is on.
- Previous Route’s **Filter** expression evaluates to true, (for example, `true`, `1 === 1`).

Note that the third condition above can be triggered intermittently by a randomizing method like `Math.random()`. This might be included in a previous Route’s own Filter expression, or in a Pipeline Function (such as one configured for random data sampling).

Route With the Output Router

Output Router Destinations offer another way to route data. These function as meta-destinations, in that they allow you to send data to multiple peer Destinations based on rules. Rules are evaluated in order, top->down, with the first match being the winner.

Choose Between QuickConnect, Routes, and Output Router

Cribl Edge offers multiple ways to route data to Destinations. Choose based on your filtering and routing requirements:

Method	Best For	Key Characteristics
QuickConnect	Rapid prototyping; parallel paths without Route-level filtering	Bypasses Routes entirely. Can connect one Source to multiple Destinations with different Pipelines, but there is no filter-based routing at the Route level.
Routes	Filter-based routing; cloning subsets of events to different Destinations	Applies filter expressions to select event subsets. Use **Final** toggle to control cloning behavior.
Output Router	Content-based fan-out from a single Route	Evaluates rules against event content after Pipeline processing. Enables conditional routing to multiple Destinations from one Pipeline.

None of these methods isolates a Source from the effects of a blocked Destination. Backpressure behavior is determined by the configuration of each Destination, not by the routing construct. If one downstream Destination blocks, it can prevent the same input data from reaching other Destinations, unless the blocking Destination uses Persistent Queue with available capacity.

For more information, see Backpressure Behavior with Multiple Destinations.

14.4. Pipelines

Data matched by a given Route is delivered to a Pipeline. Pipelines are the heart of Cribl Edge processing. Each Pipeline is a list of Functions that work on the data.

As with Routes, the order in which the Functions are listed matters. A Pipeline's Functions are evaluated in order, top->down.



Consider using Cribl Copilot Editor for help building data Pipelines. Instead of building a Pipeline manually, you can describe what you want to do with your data and Copilot Editor will generate a working Pipeline for you. See Create Pipelines Using Cribl Copilot Editor for more information.

Access Pipelines

To access Pipelines:

1. In the sidebar, select **Fleets**, then select a Fleet.
2. On the **Fleets** submenu, select **More**, then **Pipelines**.



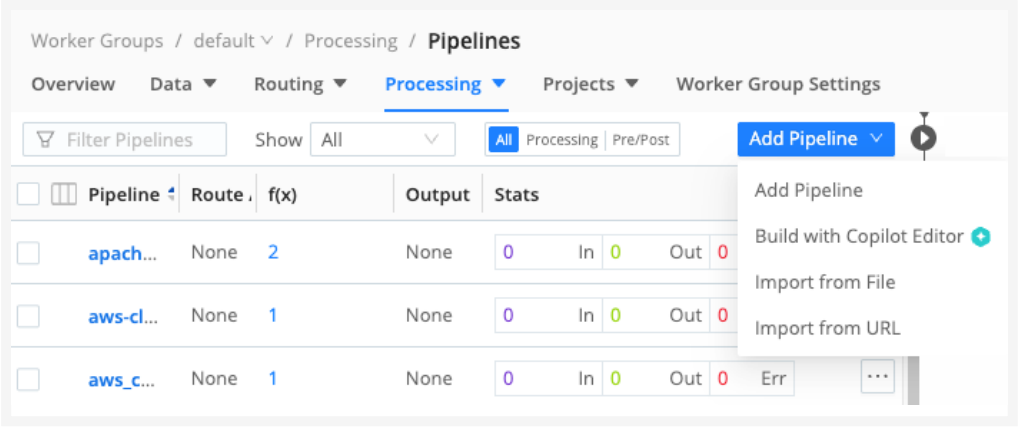
On a Single-instance deployment, select **More** straight from the **Home** submenu.

After you've opened a Pipeline, the right Preview pane adds a Pipeline Status tab, which you can select to see the Pipeline's events throughput.

Add Pipelines

To create a new Pipeline, or to import an existing Pipeline to a different Cribl Edge instance, select **Add Pipeline** at the upper right. The resulting menu offers three options:

- **Add Pipeline**: Configure a new Pipeline from scratch, by adding Functions in Cribl Edge's graphical UI.
- **Add Pipeline with Cribl Copilot**: Use the Cribl Copilot Editor to create a Pipeline using AI. See Create Pipelines Using Cribl Copilot Editor to learn more.
- **Import from File**: Import an existing Pipeline from a `.json` file on your local filesystem.
- **Import from URL**: Import an existing Pipeline from `.json` file at a remote URL. (This must be a public URL ending in `.json` - the import option doesn't pass credentials to private URLs - and the target file must be formatted as a valid Pipeline configuration.)



Create or import a Pipeline

To export a Pipeline, see Advanced Mode (JSON Editor).

To import or export a Pipeline along with broader infrastructure (like Knowledge Objects and/or sample data files), see Packs.

Default Pipelines

Cribl Edge ships with a set of built-in default Pipelines. These include:

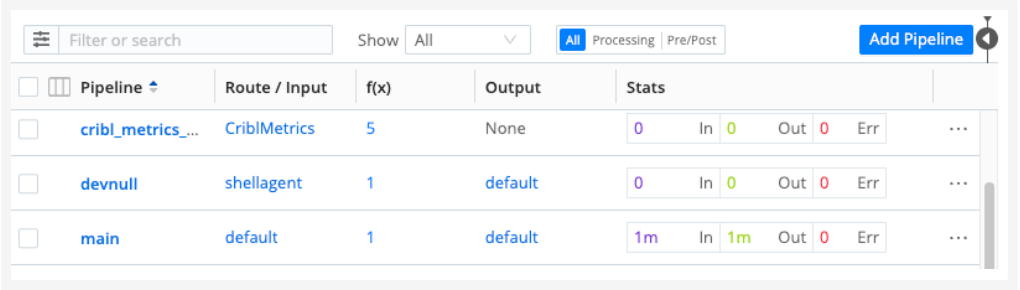
- `devnull`
- `main`
- `passthru`

These Pipelines are managed by Cribl. Cribl Edge re-creates them from its default configuration when you install or upgrade Edge, create new Fleets, or seed configuration from the built-in Packs. As a result, they cannot be permanently deleted. Removing them in the UI only affects the current configuration copy, and they will reappear the next time defaults are applied.

For production, we recommend building your own Pipelines or using Pipelines provided by Packs, and simply not attaching the default Pipelines to any Sources, Routes, or Destinations you don't intend to use.

How Pipelines Work

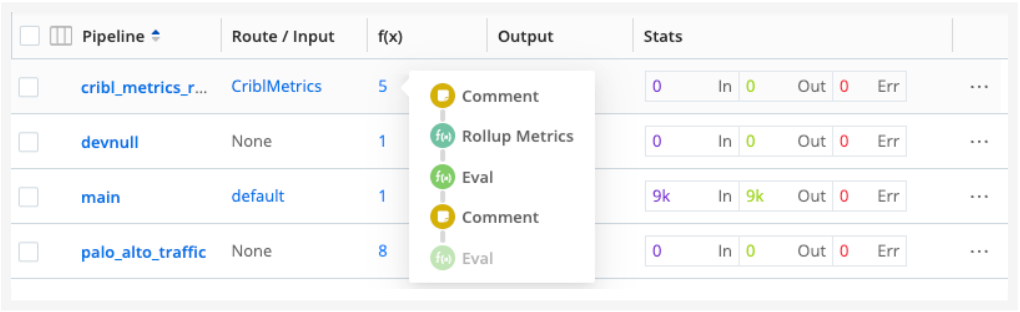
Events are always delivered to the beginning of a Pipeline via a Route. The data in the **Stats** column shown below are for the last 15 minutes.



Pipelines and Route inputs

You can press the] (right-bracket) shortcut key to toggle between the Preview pane and an expanded Pipelines display. (This shortcut works when no field has focus.)

In the condensed Pipelines display above, you can also hover over any Pipeline's **Functions** column to see a vertical preview of the stack of Functions contained in the Pipeline:



Preview on hovering over the top Pipeline (highlighted in gray)

Within the Pipeline, events are processed by each Function, in order. A Pipeline will always move events in the direction that points outside of the system. This is on purpose, to keep the design simple and avoid potential loops.



Pipeline Functions

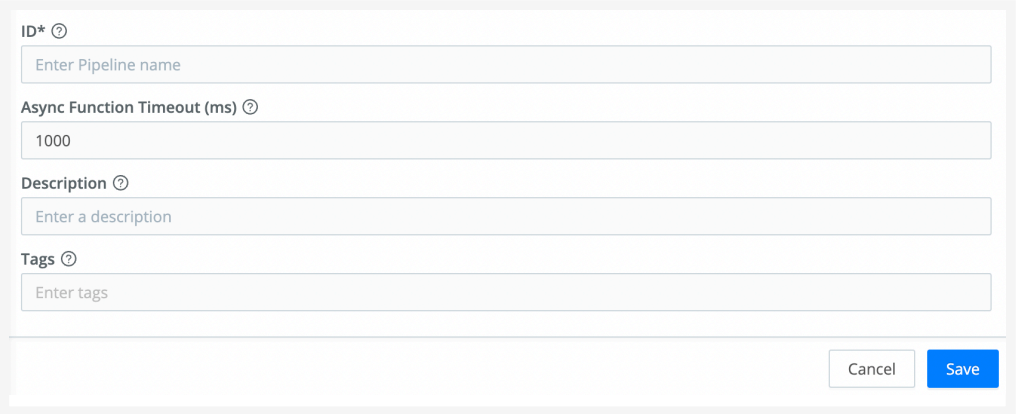
Use the **Attach to Route** link at the top to associate a new Pipeline with a Route.

> You can streamline a complex Pipeline's display by organizing related Functions into Function groups.

Pipeline Settings

Select the gear button at the top right to open the Pipeline's Settings. Here, you can:

- Use the **Async function timeout (ms)** to set the maximum amount of processing time, in milliseconds, that a Function is allowed to take before it is terminated. This prevents a Function from causing undesirable delays in your Pipeline (for example, a Lookup Function taking too long to process a large lookup file).
- Use the **Tags** field to attach arbitrary labels to the Pipeline. Once attached, you can use these tags to filter/search and group Pipelines.



Pipeline Settings

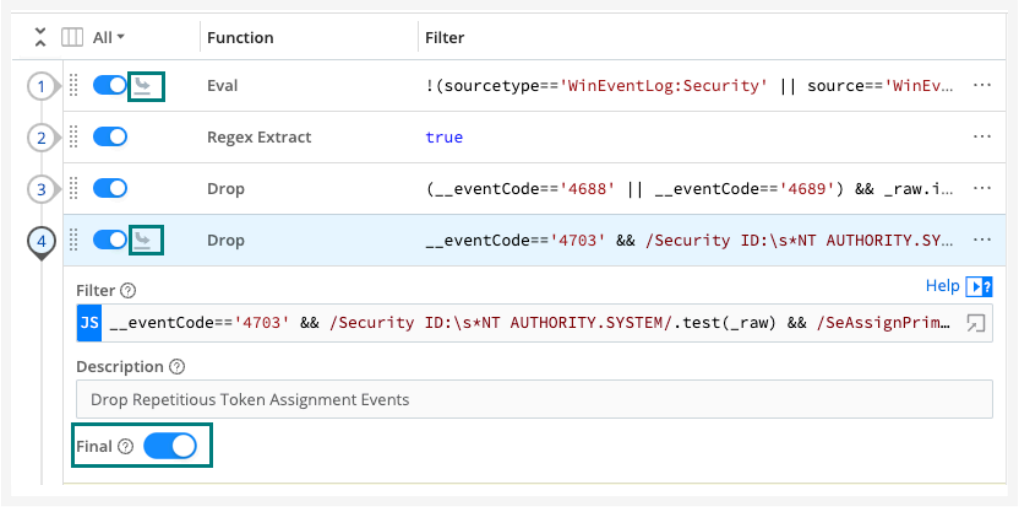
The Final Toggle

The **Final** toggle in Function settings controls what happens to the results of a Function.

Toggle **Final** off (default) to process events by this Function, **and** then pass them on to the next Function below.

Toggle **Final** on if you want the Function to "consume" the results, meaning they will not pass down to any other Function below.

An icon in Pipeline view indicates that a Function is **Final**.



The Eval and first Drop Functions bearing the Final icons

Advanced Mode (JSON Editor)

Once you've selected the gear button to enter Pipeline Settings, you can select **Manage as JSON** at the upper right to edit the Pipeline's definition in a JSON text editor. In this mode's editor, you can directly edit multiple values. You can also use the **Import** and **Export** buttons here to copy and modify existing Pipeline configurations, as `.json` files.



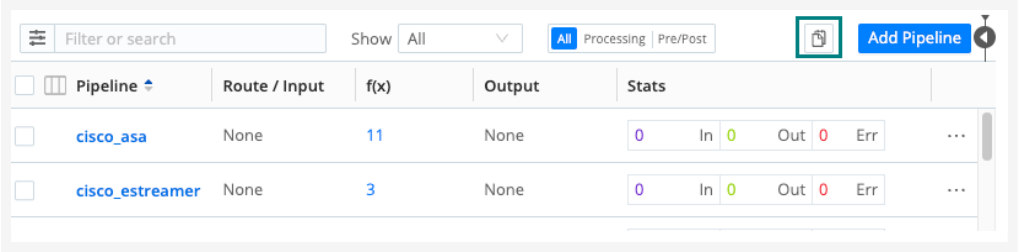
Advanced Pipeline Editing

Select **Edit in GUI** at upper right to return to the graphical Pipeline Settings page; then select **Back to <pipeline-name>** to restore the graphical Pipeline editor.

Pipeline Actions

Select a Pipeline's Actions (**...**) menu to display options for copying or deleting the Pipeline.

Copying a Pipeline displays a confirmation message and the (highlighted) Paste button shown below.



Paste button for copied Pipeline

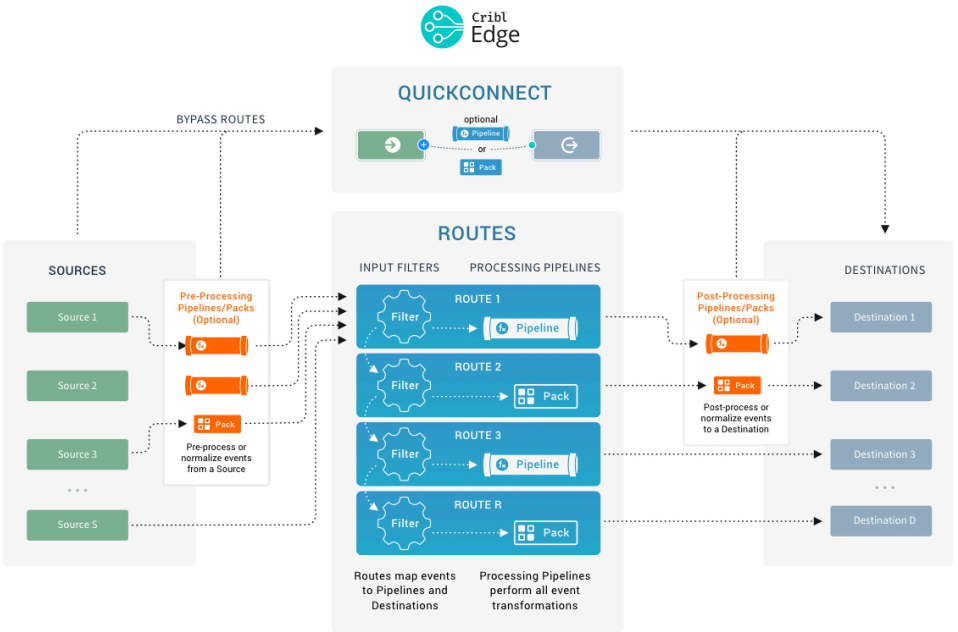
Pasting prompts you to confirm, or change, a modified name for the new Pipeline. The result will be an exact duplicate of the original Pipeline in all but name.

Chain Pipelines

You can use the Chain Function to send the output of a Pipeline to another Pipeline or Pack. There are scope restrictions within Packs, and general guardrails against circular references.

Types of Pipelines

You can apply various Pipeline types at different stages of data flow. All Pipelines have the same basic internal structure (a series of Functions) - the types below differ only in their position in the system.



Pre-processing, processing, and post-processing Pipelines

Pre-Processing Pipelines

These are Pipelines that are attached to a Source to condition (normalize) the events **before** they're delivered to a processing Pipeline. They're optional.

Typical use cases are event formatting, or applying Functions to **all** events of an input. (For example, to extract a `message` field before pushing events to various processing Pipelines.)

You configure these Pipelines just like any other Pipeline, by selecting **More** > **Pipelines** from the **Fleets** submenu. You then attach your configured Pipeline to individual Sources, using the Source's **Pre-Processing** > **Pipeline** drop-down.

Fields extracted using pre-processing Pipelines are made available to Routes.

Processing Pipelines

These are "normal" event processing Pipelines, attached directly to Routes.

Post-Processing Pipelines

These Pipelines are attached to a Destination to normalize the events before they're sent out. A post-processing Pipeline's Functions apply to **all** events exiting to the attached Destination.

Typical use cases are applying Functions that transform or shape events per receiver requirements. (For example, to ensure that a `_time` field exists for all events bound to a Splunk receiver.)

You configure these Pipelines as normal, by selecting **More** > **Pipelines** from the **Fleets** submenu. You then attach your configured Pipeline to individual Destinations, using the Destination's **Post-Processing** > **Pipeline** drop-down.

You can also use a Destination's **Post-Processing** options to add **System Fields** like `cribl_input`, identifying the Cribl Edge Source that processed the events.

Best Practices for Pipelines

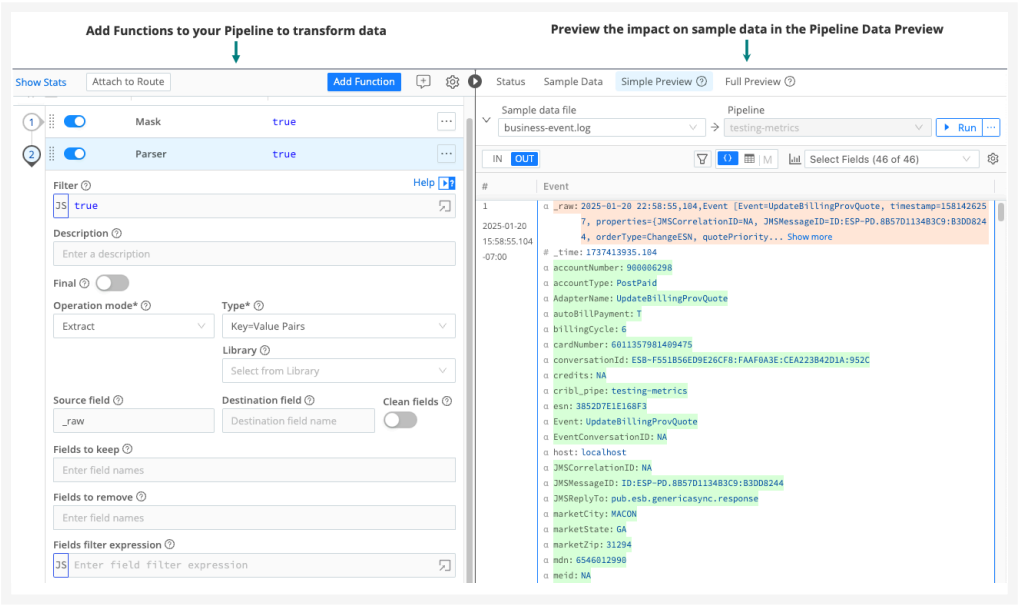
Functions in a Pipeline are equipped with their own filters. Even though filters are not required, we recommend using them as often as possible. See Build Custom Logic to Route and Process Your Data: Filter Expressions for more information.

14.4.1. Validate Pipeline Logic Using Data Preview

You can use the Data Preview tool to preview, test, and validate the Functions you add to a Pipeline or Route. It helps ensure that your Pipeline and Function configurations produce the expected output, giving you confidence that Cribl Edge processes your data as intended.

The Data Preview tool works by processing a set of sample events and passing them through the Pipeline. Then, it displays the inbound and outbound results in a dedicated pane. Whenever you modify, add, or remove a Function, the output updates instantly to reflect your changes.

For example, suppose you add a Parser Function to your Pipeline to convert raw, unstructured data into structured key-value pairs. After parsing, you may choose to promote these key-value pairs to the field level to enhance data processing and analysis. To verify these changes meet your expectations, you can use the **OUT** tab in the Data Preview pane to review the transformed data.



Add Functions to your Pipeline to transform data, then preview the impact on sample data in Data Preview

In addition to previewing data, the Data Preview offers:

- **Metrics View**: This metrics-first view in the Data Preview pane automatically detects and visualizes metrics in your sample data, organizing them by metric name for a more intuitive experience. Designed for metrics-heavy datasets, this view provides insights into key metrics, dimensions, and time series. It helps you identify high-cardinality data (datasets with a large number of uncommon or unique values), enabling you to make informed decisions about data processing before forwarding it to potentially costly downstream receivers. See Metrics View and Manage Metrics and High Cardinality for more information.
- **Pipeline diagnostics**: Includes status updates and summary throughput statistics to assess the health of the Pipeline and its Functions. See Pipeline Diagnostics and Statistics for more details.

Data preview on the **Simple Preview** tab is limited to 10 MB of data. This cap prevents excessive data processing and safeguards system stability. If the dataset is more than 10 MB when a simple preview runs, Cribl Edge will limit the preview to 10 MB and stop processing data beyond that limit.



The Get Started With Cribl Stream tutorial provides hands-on experience with the Pipeline Data Preview if you would like to learn more about this tool through personal experimentation and exploration.

How to Use the Data Preview Tool

To use the Data Preview tool:

1. Create or import some sample data. See Create and Share Data Samples for information about different methods to generate sample data.
2. Then, use the different Data Preview views to test and validate the changes you make in your Pipeline. See Explore Data and Validate Pipelines with Different Views to learn about the different views you can apply to your data.

Explore Data and Validate Pipelines with Different Views

Once you have created or imported sample data for your Pipeline, you can explore and analyze it using the various views available in the Data Preview pane. These views allow you to preview and test changes in real time, helping ensure that your data processing Functions produce the expected results. By leveraging the Data Preview, you can confidently validate that your Pipeline transformations align with your intended outcomes.

Simple or Full Previews

Select **Simple Preview** or **Full Preview** beside a file name to display its events in the Preview pane:

- **Simple Preview**: View events on either the **IN** or the **OUT** (processed) side of a single Pipeline.
- **Full Preview**: View events on the **OUT** side of either the processing or post-processing Pipeline. Selecting this option expands the Preview pane's upper controls to include an **Exit Point** drop-down, where you make this choice.

Before and After Views

The Data Preview pane offers two time-based display options for previewing sample data:

- The **IN** tab displays samples as they appear on the way *in* to the Pipeline, before data processing.

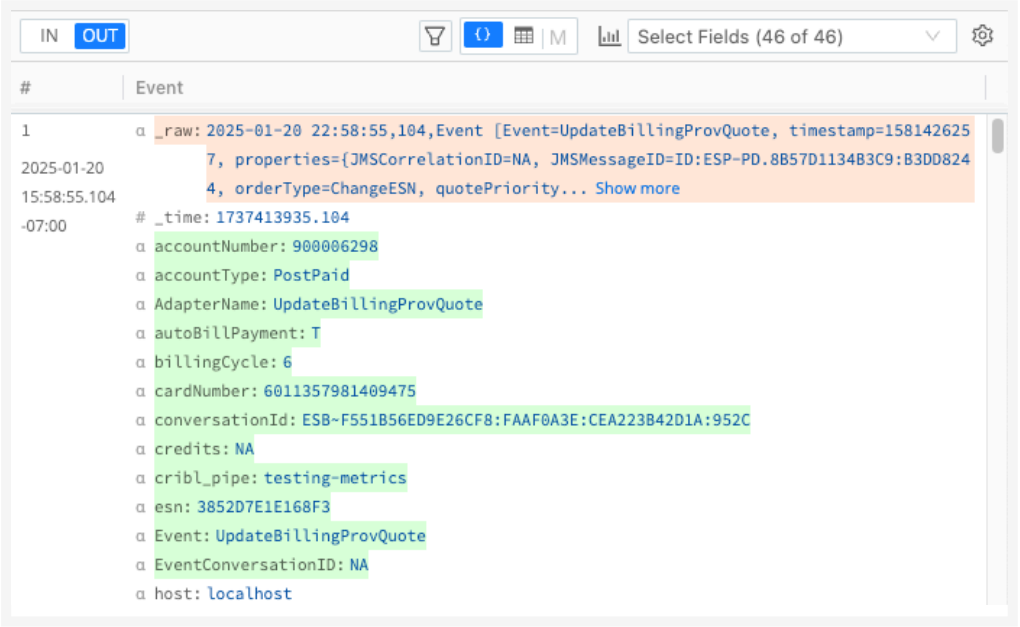
- The **OUT** tab displays the same information, but shows the sample data as it will appear on way *out* of the Pipeline after the data processing. Use the **Select Fields** drop-down menu to control which fields the **OUT** tab displays.

In most cases, you will typically use the **OUT** tab to preview, test, and validate your Pipeline.

Event and Table Views

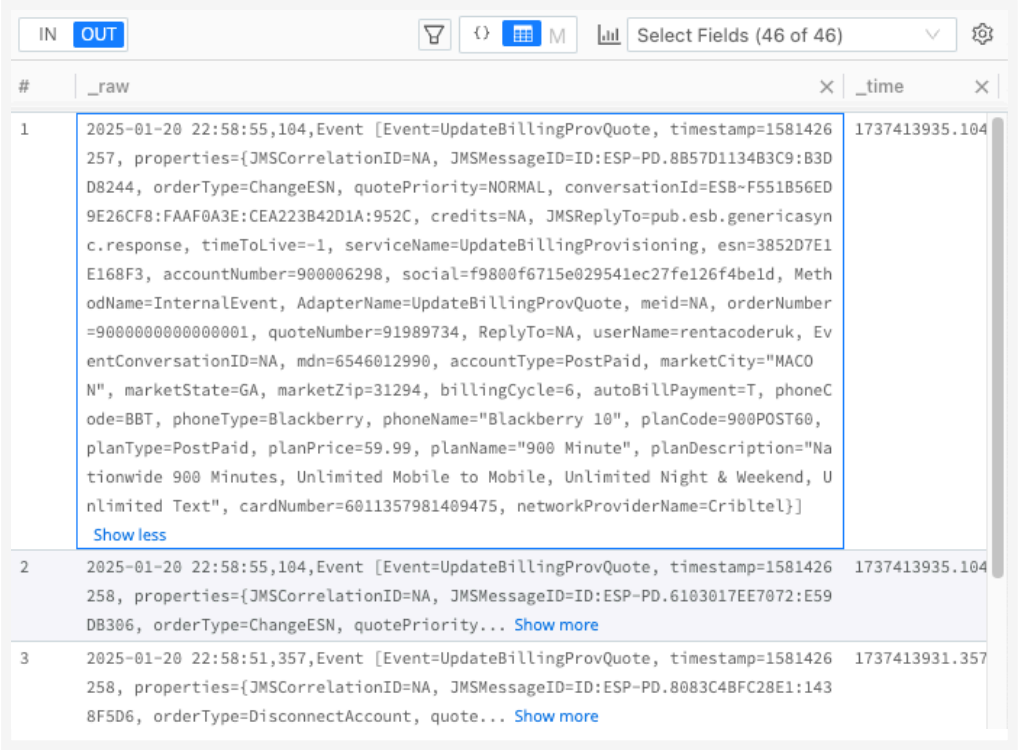
The Data Preview toolbar has several available views. Each format can be useful, depending on the type of data you are previewing.

The Event View displays data in JSON format for both the **IN** and **OUT** tabs.



Example Event View

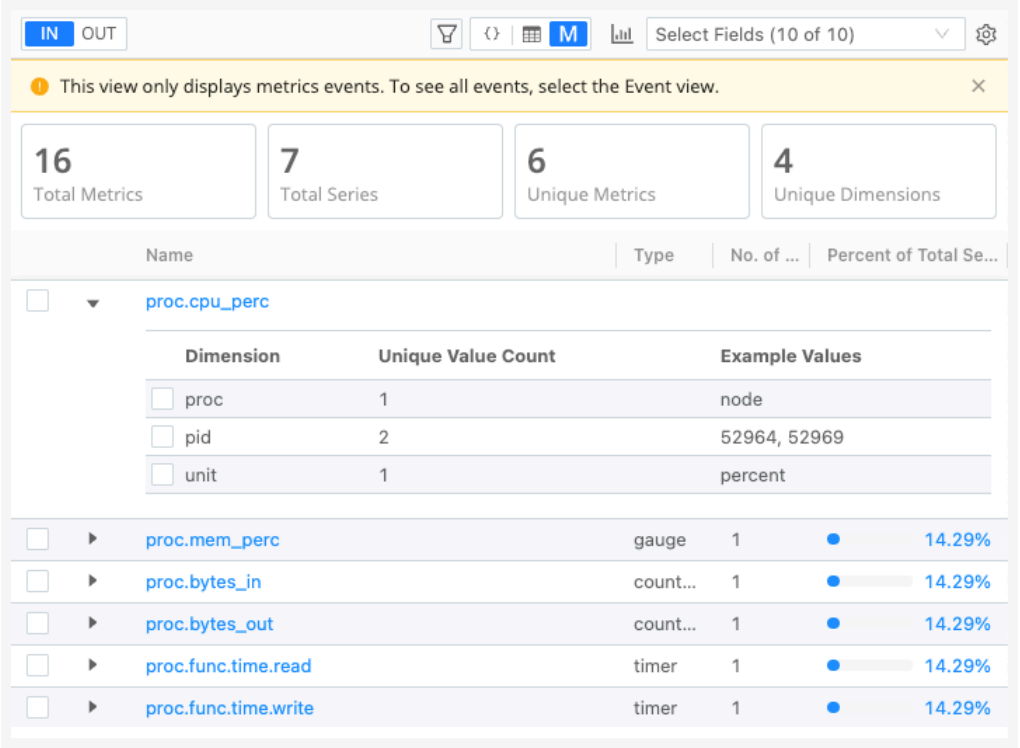
The Table View in displays data as a table for both the **IN** and **OUT** tabs.



Example Table View

Metrics View

If your data sample contains metric events, the Data Preview automatically switches to the Metrics view. It intelligently detects the underlying data structure and switches to this view when needed to display relevant insights. This metrics-first view allows you to visualize metric events as you build your Pipeline. It allows you to analyze metrics, dimensions, time series, the number of unique values, and the percentage each metric contributes to the total time series. See Manage Metrics and High Cardinality for more information about using Metrics View.



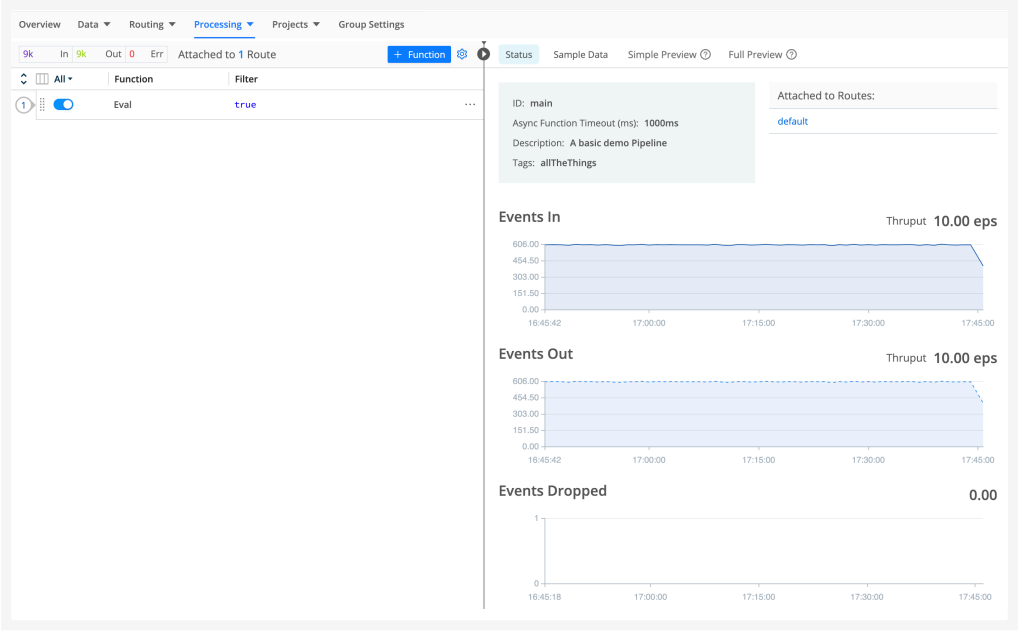
Example Metrics View

Pipeline Diagnostics and Statistics

This section explains the available tools in the Data Preview that you can use to get information about your Pipeline and Functions.

Check Pipeline Status

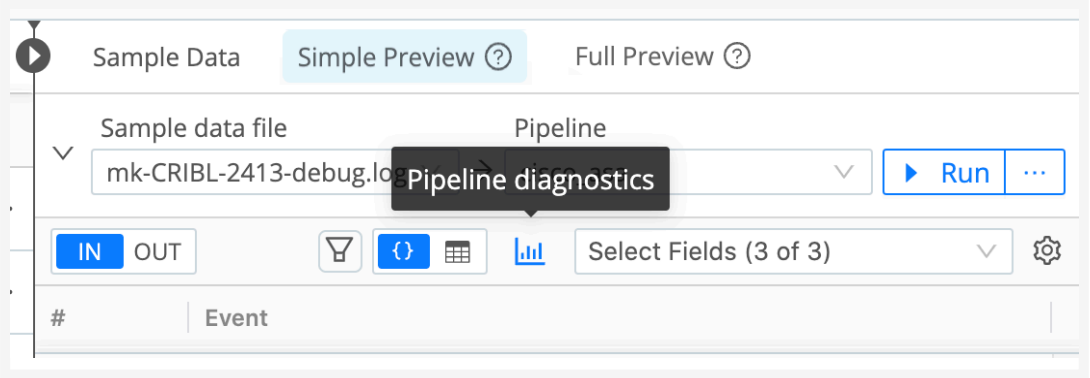
From the Pipelines page, select any Pipeline to display a **Status** tab in the Data Preview pane. Select this tab to show graphs of the Pipeline **Events In**, **Events Out**, and **Events Dropped** over time, along with summary throughput statistics (in events per second).



Pipeline status tab

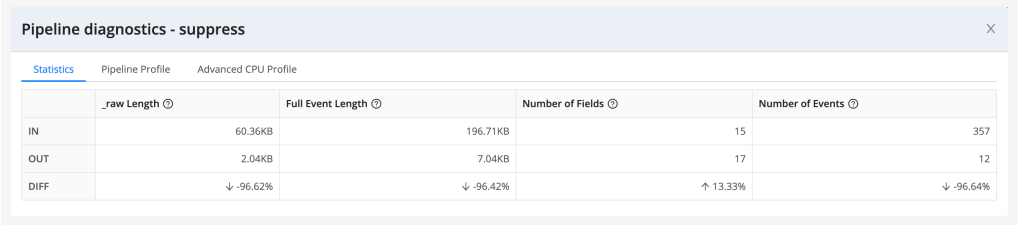
Pipeline Profiling

At top of the Preview pane, you can select a sample data file and Pipeline, then select the Pipeline diagnostics (bar-graph) button to access statistics on the selected Pipeline's performance.



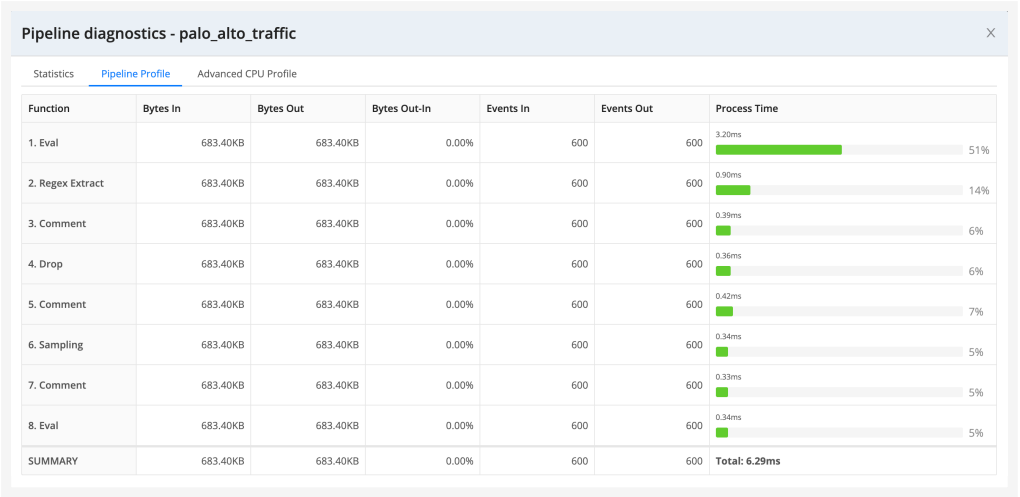
Pipeline diagnostics

The **Statistics** tab in the resulting modal displays the processing impact of the Pipeline on field length and counts.



Pipeline diagnostics > Statistics tab

The **Pipeline Profile** tab, available from the **Simple Preview** tab, helps you debug your Pipeline by showing the contribution of individual Functions to data volume, events count, and the Pipeline processing time. Every Function has a minimum processing time, including Functions that make no changes to event data, such as the Comment Function.

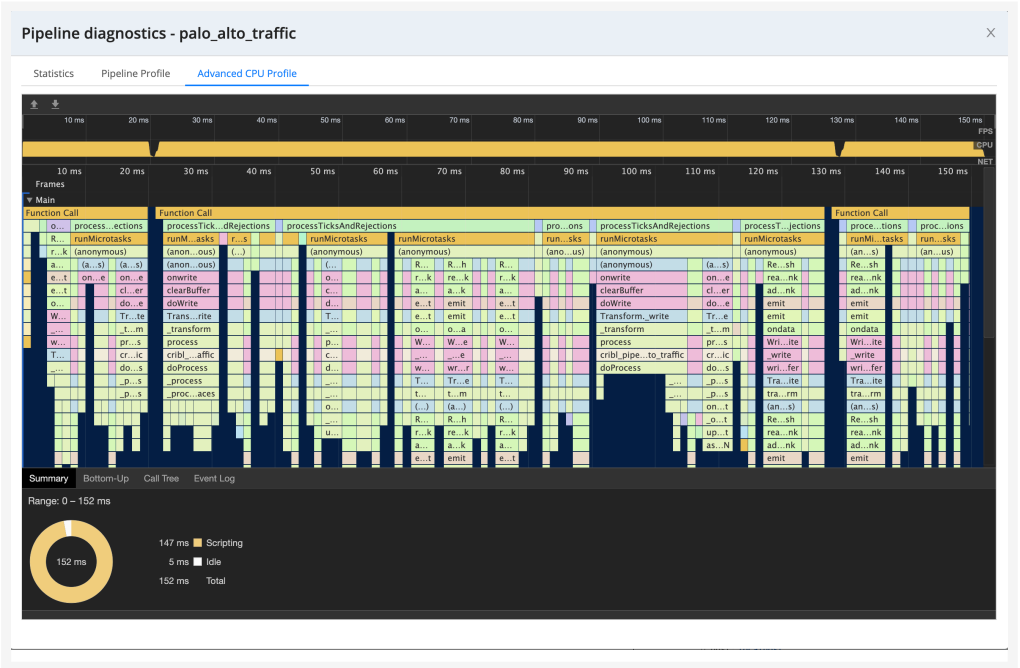


Function	Bytes In	Bytes Out	Bytes Out-In	Events In	Events Out	Process Time
1. Eval	683.40KB	683.40KB	0.00%	600	600	3.20ms 51%
2. Regex Extract	683.40KB	683.40KB	0.00%	600	600	0.90ms 14%
3. Comment	683.40KB	683.40KB	0.00%	600	600	0.39ms 6%
4. Drop	683.40KB	683.40KB	0.00%	600	600	0.36ms 6%
5. Comment	683.40KB	683.40KB	0.00%	600	600	0.42ms 7%
6. Sampling	683.40KB	683.40KB	0.00%	600	600	0.34ms 5%
7. Comment	683.40KB	683.40KB	0.00%	600	600	0.33ms 5%
8. Eval	683.40KB	683.40KB	0.00%	600	600	0.34ms 5%
SUMMARY	683.40KB	683.40KB	0.00%	600	600	**Total: 6.29ms**

Pipeline diagnostics > Pipeline Profile tab

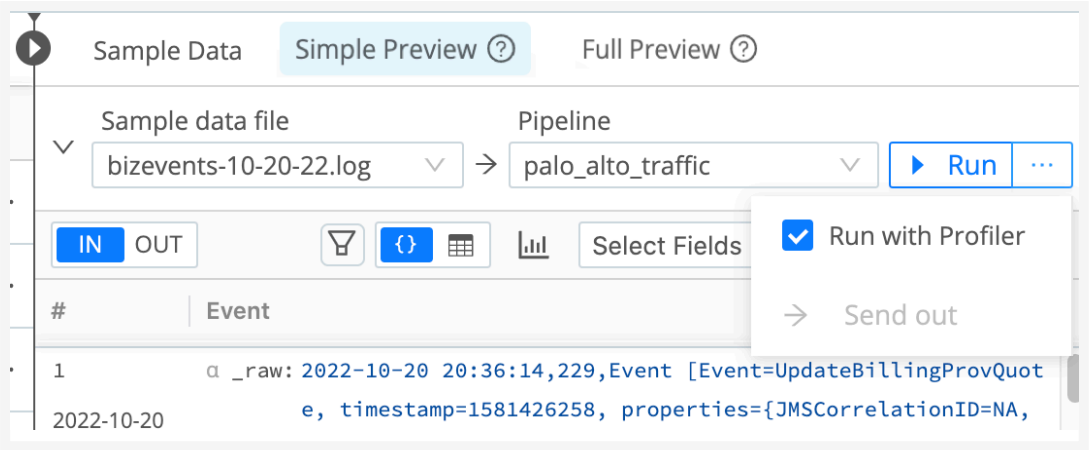
When you profile a Chain Function, this tab condenses all of the chained Pipeline's processing into a single **Chain** row. To see individual statistics on the aggregated Functions, profile the chained Pipeline separately.

The **Advanced CPU Profile** tab, available from the **Simple Preview** tab, displays richer details on individual Function stacks.



Pipeline diagnostics > Advanced CPU Profile tab

You can disable the profiling features to regain their own (minimal) resources. To disable profiling features, select the drop-down list beside the **Run** button and clear the **Run with Profiler** check box.



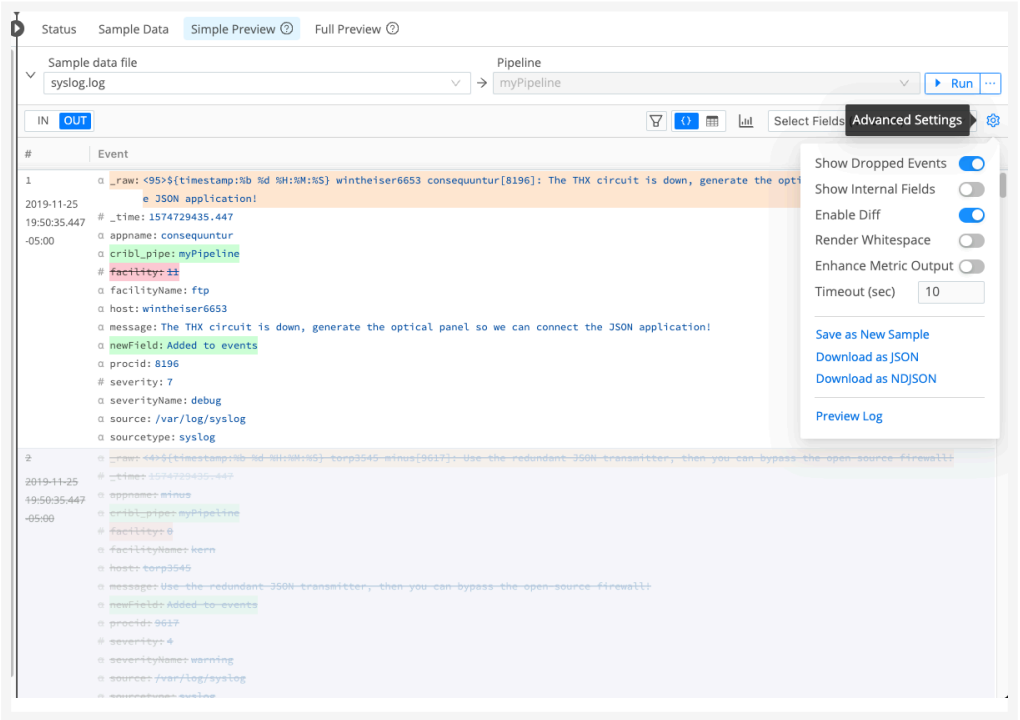
Disabling Pipeline profiling

Advanced Settings

The **Advanced Settings** menu ⚙ includes **Show Dropped Events**, **Show Internal Fields**, and **Enable Diff** toggles, which control how the **OUT** tab displays changes to event data.

If you enable the **Show Dropped Events** and **Enable Diff** toggles, the **OUT** tab displays event data as follows:

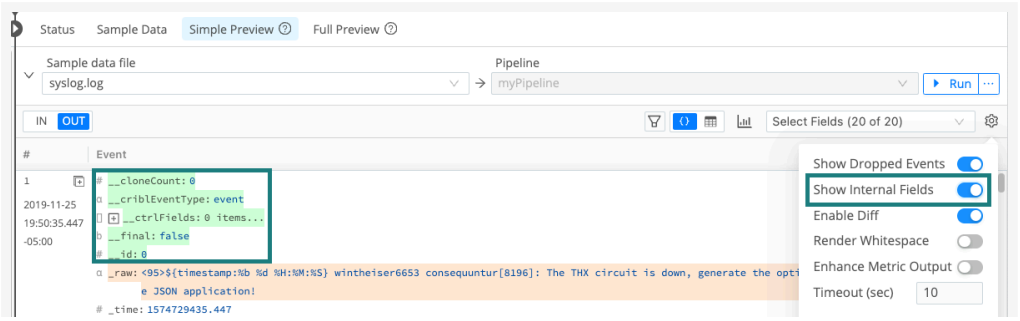
- Modified fields are highlighted in amber.
- Added fields are highlighted in green.
- Deleted fields are highlighted in red and displayed in strikethrough text.
- Dropped events are displayed in strikethrough text.



Highlighted fields in Pipeline output

The rest of this section describes the additional fields available in **Advanced Settings**.

Show Internal Fields: Display fields that Cribl Edge adds to events, as well as Source-specific fields that Cribl Edge forwards from upstream senders.



14.5. Process Data With AI Assistance

Cribl Copilot Editor is an interactive, AI-powered assistant that helps you build data Pipelines in Cribl Edge using plain-language prompts. It is useful for anyone who wants help building Pipelines in Cribl Edge, especially if you're just getting started or working through a complex use case.

You might find Copilot Editor especially helpful if:

- **You are new to Cribl Edge and want a guided way to learn how it works.** Copilot Editor acts as an interactive learning tool. Copilot Editor is especially helpful when you're not sure how to start building a Pipeline, or when you're stuck writing complex Regex or JavaScript expressions. It can generate example Pipelines based on your goals, suggest alternative approaches, and help you explore solutions you might not have considered. Even if you don't use the generated Pipeline, it's a great way to learn how different Functions work together and how to build your own Pipelines.
- **You've inherited a Cribl deployment and need to make changes quickly.** If you're a system administrator or operator who needs to stand up a deployment fast or maintain an existing one, Copilot Editor can help you make quick progress without needing to dive deep into configuration syntax right away.
- **You want to experiment with AI to speed up Pipeline building.** Whether you're curious about AI or looking for a faster way to translate your intent into working logic, Copilot Editor lets you test ideas and see results with minimal setup.

You're always in control of the final result and can review, refine, or copy its suggestions into the Pipelines you build manually.

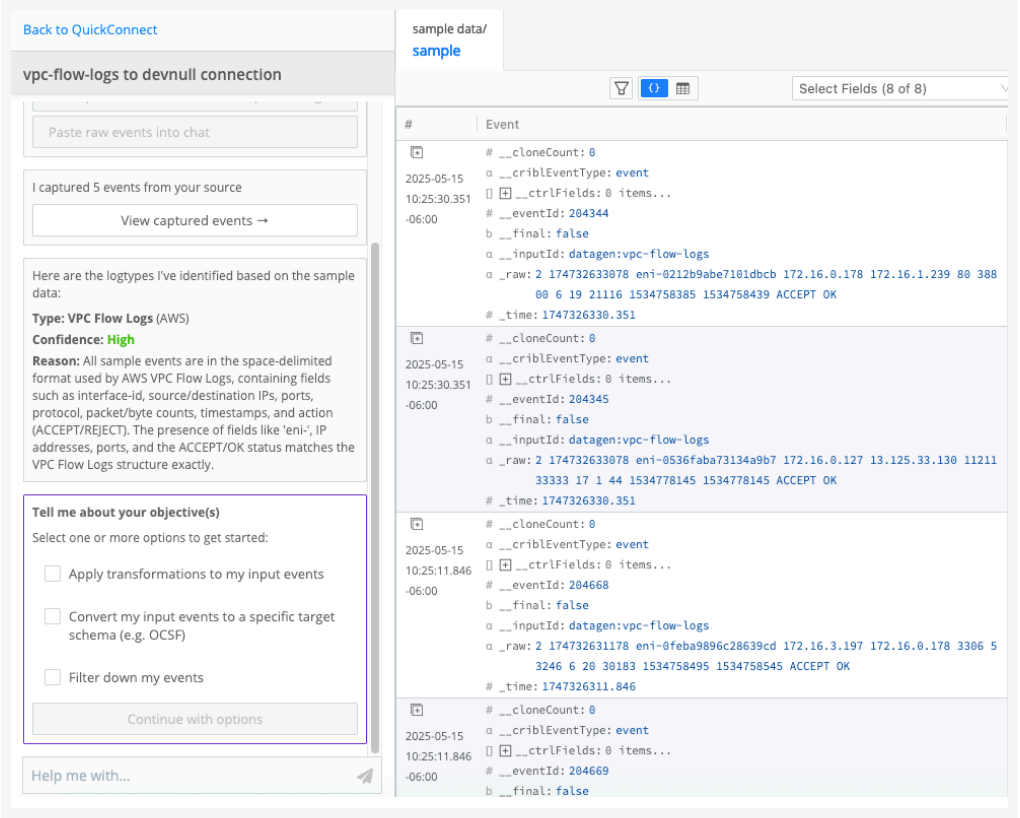
14.5.1. Create Pipelines With Cribl Copilot Editor

Cribl Copilot Editor is an interactive, AI-powered assistant that helps you build data Pipelines in Cribl Edge using plain-language prompts. Instead of building a Pipeline manually, you can describe what you want to do with your data and Copilot Editor will generate a working Pipeline for you.

Copilot Editor gives you an easy way to explore what Cribl Edge can do, especially if you're new to the platform or want help tackling more complex use cases. You stay in control: Copilot Editor's suggestions are editable, and you can review, modify, or view outputs from the suggestions before using them in production.

Here are a few common ways you can use Copilot Editor:

- **Structure raw data to match a standard schema or convert from one schema to another:** The data for many Destinations follow specific schemas. If events are improperly formatted, some Destinations may drop data or store them in ways that make the data difficult to find and use. With Cribl Edge, you can convert your raw data into normalized schemas, using Copilot to do the heavy lifting around configuring Functions. Copilot Editor is optimized for transforming data into the OCSF, UDM, and ECS schemas. You can also upload your own schema objects to convert to other schemas. See Convert Data into a Custom Schema for information about using custom schemas or schemas that aren't yet officially supported.
- **Apply common Functions to transform or enrich your data:** Copilot Editor can apply Cribl Edge's most frequently used Functions based on your intent. You don't need to know Regex or JavaScript syntax or which Functions to use up front. Just describe your goals and Copilot Editor builds the appropriate logic for you.
- **Filter and route data dynamically:** Use plain-language prompts to define filtering or routing logic without writing Regex or JavaScript expressions manually. Copilot Editor builds the first draft of the filtering and control logic. You can use its suggestions as-is or fine-tune as needed.



Copilot Editor

How to Use Copilot Editor

This section explains the overall workflow for using Copilot Editor to build Pipelines.

Before You Begin

Before you can start using Copilot Editor, you must first enable Cribl Copilot for your Organization:

1. From **Settings**, select the **Global** tab, and then select **AI Settings**.
2. Read the privacy policy and accept the terms and conditions.



See Enable or Disable Cribl AI Features for information about how to enable or disable Cribl Copilot programmatically.

Launch Copilot Editor

You can launch Copilot Editor in three ways:

- Through QuickConnect.

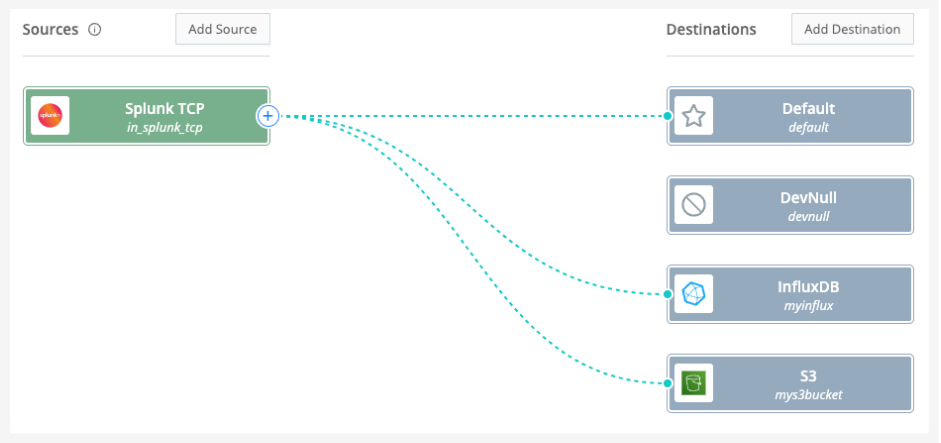
- Through the Pipelines page.
- Inside a Pipeline through the **Add Function** menu.

Select a tab for more detailed instructions about each method for launching Copilot Editor:

QuickConnect | Pipeline page | Add Function menu

To launch Copilot Editor using QuickConnect, first set up your QuickConnect environment:

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Worker Groups**, select a Worker Group.
2. Select the **QuickConnect** tile.
3. Select **Add Source** to add at least one Source and **Add Destination** to add at least one Destination. Configure them as needed. Any previously configured Sources and Destinations automatically appear in QuickConnect.
4. Connect a Source to a Destination, but do not create a Pipeline yet.



Connect a Source to a Destination

5. Select a connection to open the **Connection Configuration** modal. Select **Build with Copilot Editor** to launch the interactive Copilot Editor modal.

You can also select **Build with Copilot Editor** at the top of the **QuickConnect** page to launch.

Build a Pipeline with Copilot Editor

The interactive flow of Copilot Editor follows a few general phases:

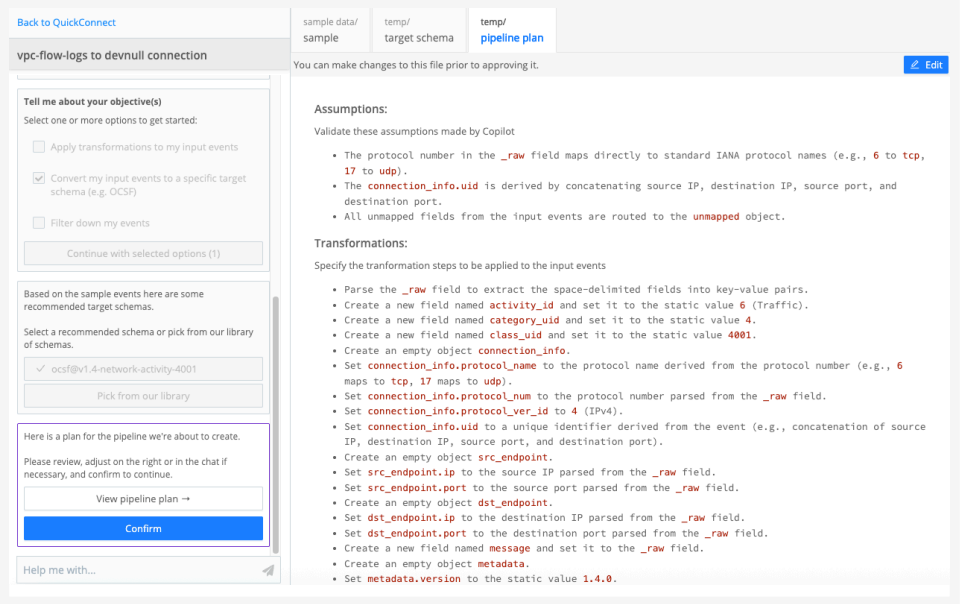
1. **Capture sample data:** When you first launch Copilot Editor, you see a chat window that prompts you to capture sample data. It asks you to choose a method for capturing sample data, such as collecting a live sample or by uploading sample data for that Source.

Why does it need sample data?

> Copilot Editor uses this data to help you build, preview, test, and validate any data transformations you ask it to create in your Pipeline. It helps you compare the inbound and outbound data to ensure that any Pipeline and Function configurations created with Copilot Editor produce the expected output, giving you confidence it is processing your data as intended.

2. **Choose an objective:** After analyzing your sample data, Copilot Editor displays your data in the preview window, similar to how data is presented in Data Preview in a standard Pipeline. The chat window will summarize the data and present you with a set of options for transforming your data. Select any objectives you would like to apply to your data. Copilot Editor will prompt you if it needs additional information about your objectives.
3. **View and edit the Pipeline plan:** Based on your selection, Copilot Editor generates a suggested Pipeline plan. It outlines:
    - Assumptions it made when generating the plan.
    - Transformations that it recommends based on your objectives.
    - Fields to keep and drop.

> If needed, select **Edit** to further customize the proposed plan before confirming. For example, you could rename any fields as needed.



Copilot Editor Pipeline Plan

4. **Confirm the Pipeline plan and build the Pipeline:** After you confirm the plan, Copilot Editor builds a suggested Pipeline and displays a preview. Expand the Functions to explore the settings it applied. You can continue to edit the Pipeline at this stage as needed. You can also use natural language prompts in the chat window to request changes.
5. **Apply the Pipeline:** After you apply the Pipeline, select **Back to QuickConnect** at the top of the chat window to return to QuickConnect. Select **Save Connection** to add the Pipeline to your deployment.



Take note of the Pipeline ID displayed in the chat window so that you can find this Pipeline later.

6. **Confirm the Pipeline is present:** Confirm that the Pipeline ID you built with Copilot Editor appears in the Pipeline page or in QuickConnect. To find it in QuickConnect, select the connection that you used to launch Copilot Editor or its Pipeline icon, then select **Pipeline** to view the list of Pipelines for this connection.

Edit or Update a Pipeline With Copilot Editor

Any time you run a Copilot Editor, it eventually asks you to choose an objective for that session. You can select **Edit an Existing Pipeline** as the objective. It then presents you with a list of Pipelines to select and edit.

With that in mind, you can start a Pipeline editing session with Copilot Editor from a few different entry points:

- Start a new Copilot Editor session, then select the **Edit an Existing Pipeline** as the objective.
- Open an existing Pipeline, select **Add Function** , then **Edit with Copilot Editor**. Select the **Edit an Existing Pipeline** as the objective.



You can edit any existing Pipeline with Copilot Editor, not just Pipelines that were previously generated with Copilot Editor.

Guidelines and Better Practices

When working with Copilot Editor, keep these guidelines in mind:

- **Always review and validate Pipeline suggestions:** As with any AI tool, you should always check and confirm that the suggested changes match your expectations. Trust, but verify.
- **Copilot Editor may take different paths to meet an objective:** AI tools do not always solve problems in the exact same way every time, which sometimes takes some getting used to. This behavior is by design, but just be aware that the results may vary.

Use effective prompts to ensure the best results from the Copilot Editor. Some general guidelines:

- **Be specific:** To receive high-quality results, be clear and explicit with your requests. Provide exact names of fields that you want to reference or create.
- **Provide examples:** For instance, if you want to mask a specific piece of text, describe the target text in detail and explain what it should be masked to (for example, `Mask all IPv4 IP addresses to X.X.X.X`).
- **Ask questions:** Use the chat window to answer general questions about Pipelines. For example, you could ask the chatbot, `What Pipeline Functions can I use to redact sensitive`

```
information like IP addresses?
```

Feature Support

Cribl does not make any warranty regarding the availability, uptime, performance, or accuracy of Cribl Copilot, including PII Detection and Prevention. Please read the Supplemental Terms for Cribl Copilot for more information.

Requirements

Cribl Copilot Editor always requires internet access.

Supported Deployment Types

Cribl Copilot Editor is available:

- For Cribl.Cloud or hybrid deployments.
- For Cribl Edge 4.12.0 and newer.

Supported Schemas

Cribl Copilot Editor specializes in transforming data into these schemas:

Schema	Supported Version(s)
OCSF Open Cybersecurity Schema Framework	• 1.6.0 • 1.5.0 • 1.4.0
UDM Unified Data Model	• 6.0
ECS Elastic Common Schema	• 9.0.0 (base version only)

Supported Functions and Integrations

Copilot Editor can build Pipelines using these Functions:

- Aggregations
- Code
- Comment
- Drop
- Eval
- Flatten
- Lookup
- Mask
- Numerify
- Regex Extract
- Regex Filter
- Rename
- Serialize
- Unroll

14.5.2. Convert Data Into A Custom Schema

Copilot Editor is an interactive AI assistant that helps you build Pipelines more easily in Cribl Edge. It specializes in designing Pipelines that transform your data to match specific schemas, whether they are industry-standard or custom-defined.

With Copilot Editor, you can:

- Structure raw data into a standard schema, such as OCSF, UDM, and ECS.
- Convert data from one standard schema to another.
- Transform data into a custom schema that you define.

Copilot Editor supports many schemas out of the box. See Supported Schemas for a complete list.

You can also define your own custom schema or adapt a standard schema for your needs. Copilot Editor can use your custom schema definition to help you generate a Pipeline that transforms your data to your specifications.

See Create Pipelines With Cribl Copilot Editor for a broader introduction, including how to launch it and build your first Pipeline.

Overview of the Custom Schema Workflow

The workflow for building a custom schema has five phases:

1. Get an example of the target schema: Start by identifying a sample event that represents your desired output format. If you already have a fully defined target schema, you can skip the first two steps and upload it directly to Cribl Edge.
2. Convert the example into JSON: Follow best practices to convert your example into a data schema using valid JSON.
3. Upload the schema: Add the JSON object to your Knowledge library so that Copilot Editor can reference that file.
4. Build a Pipeline to transform your data schema: Use Copilot Editor to create a Pipeline that can transform your data into the target schema.
5. Validate the schema output: Validate the data outputs to ensure you get the intended results. Then, make adjustments as needed.

Get an Example of the Target Schema

To build a custom data schema, start by identifying what you want your output data to look like. This step is optional if you already have a complete target schema ready to upload to Cribl Edge.

Find or create a sample event that closely matches your desired output. Choose an example that represents the type of event you want to map and reflects the ideal structure for your schema.

This guide uses the following sample event as a reference:

```
{
    "timestamp": "2025-07-08T14:30:00Z",
    "level": "INFO",
    "message": "User creation successful",
    "user": {
      "id": 12345,
      "username": "johndoe"
    },
    "session": {
      "session_id": "abcde12345",
      "device": "Chrome on Mac OS"
    }
  }
```

Copy

Convert the Example into JSON

Next, convert your sample event into a data schema using valid JSON object format. You'll upload this schema to Cribl Edge in a later step, allowing Copilot Editor to reference it when building your Pipeline.

Copilot Editor requires all schemas to be structured as valid JSON objects so it can parse and validate them correctly.

You can ensure that Copilot Editor can easily consume and successfully process your data schema by the best practices outlined in the following sections. The following example shows a well-formed JSON schema that follows best practices and serves as a reference in these sections:

```
{
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "my-custom-schema-user-creation",
  "framework": "custom",
  "description": "A custom schema used to map user creation events.",
  "type": "object",
  "additionalProperties": false,
  "properties": {
    "timestamp": {
      "type": "string",
      "description": "The timestamp of the event in RFC 3339 format."
    },
    "level": {
      "type": "string",
      "enum": ["INFO", "WARNING", "ERROR", "DEBUG", "TRACE", "CRITICAL", "ALE
      "description": "A description of the type of event."
    },
    "message": {
      "type": "string",
      "description": "A description of the event."
    },
    "user": {
      "type": "object",
      "properties": {
        "id": {
          "type": "integer",
          "description": "The ID of the user created."
        },
        "username": {
          "type": "string",
          "description": "The username of the user created."
        }
      },
      "required": ["id"]
    },
    "session": {
      "type": "object",
      "properties": {
        "session_id": {
          "type": "string",
          "description": "The ID of the session."
        },
        "device": {
          "type": "string",
          "description": "The device of the session."
        }
      },
      "required": ["session_id"]
    },
    "additional": {
      "type": "object",
      "properties": {},
      "additionalProperties": true,
      "description": "Important log data that cannot be adequately represente
```

```
55          }
56        },
57        "required": ["timestamp", "level", "user"]
58      }
```

Copy

Define the Field Names, Types, and Descriptions

Copilot Editor can map your log data more accurately if you provide meaningful context when defining each field in your schema.

For every field in your JSON schema, always include:

- The field name name of the field.
- The field's data type (such as a string, integer, or boolean).
- A brief description of the field's purpose or contents.

Here's an example illustrating these best practices, taken from the reference schema (lines 18-21):

```
1     "message": {
2       "type": "string",
3       "description": "A description of the event."
4     },
```

Copy

In this example:

- Line 1 indicates the field name: `message`.
- Line 2 defines the data type: `string`.
- Line 3 provides a description: `A description of the event.`

List Allowed Values for Restricted Fields

If a field should only accept a specific set of values, use an enum to define the allowed options. This helps Copilot Editor understand the valid inputs for that field and ensures consistent schema mapping.

For example, see this section of the reference JSON schema (lines 13-17):

```
1     "level": {
2       "type": "string",
3       "enum": ["INFO", "WARNING", "ERROR", "DEBUG", "TRACE", "CRITICAL", "ALERT",
4       "description": "A description of the type of event."
5     },
```

Copy

In this example, line 3 defines the allowed values for the level field using an `enum` array.

Mark Required Fields

Clearly define which fields are required so Copilot Editor knows which values it must always include in the generated mapping. Use a required array inside each object to mark fields that are mandatory within that scope, and add a top-level required array to indicate which fields are required across the entire schema.

For example, see this portion from of the reference schema (lines 36-58):

```
 1         "session": {
 2           "type": "object",
 3           "properties": {
 4             "session_id": {
 5               "type": "string",
 6               "description": "The ID of the session."
 7             },
 8             "device": {
 9               "type": "string",
10               "description": "The device of the session."
11             }
12           },
13           "required": ["session_id"]
14         },
15         "additional": {
16           "type": "object",
17           "properties": {},
18           "additionalProperties": true,
19           "description": "Important log data that cannot be adequately represente
20         }
21       },
22       "required": ["timestamp", "level", "user"]
23     }
```

Copy

In this example:

- Line 13 includes a required array that marks `session_id` as mandatory: `"required": ["session_id"]`.
- Line 22 is the top-level array of required fields, indicating which fields are mandatory across the entire object: `"required": ["timestamp", "level", "user"]`.

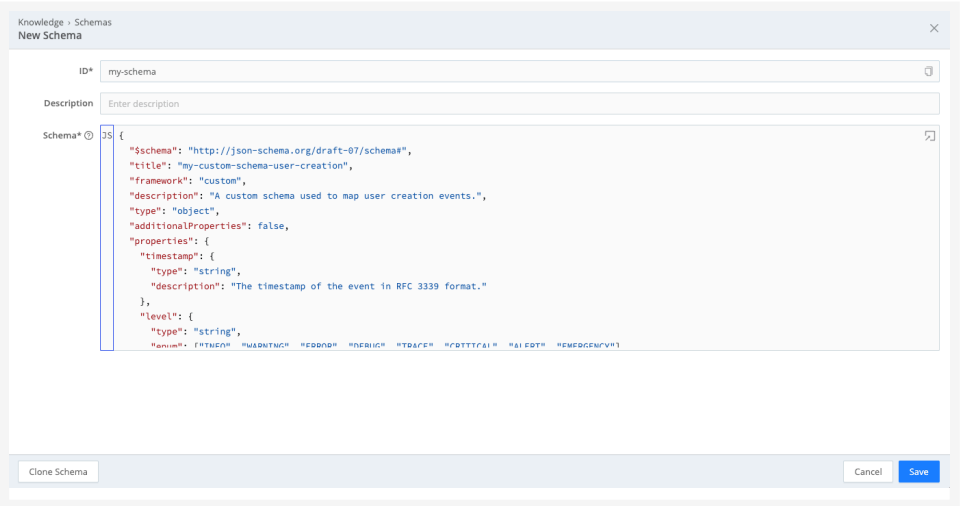
Add an Unmapped Fields Section

Consider adding an additional or unmapped field as a catch-all for any data that doesn't fit cleanly into your defined schema. This lets you retain important information from the original event without forcing it into a specific predefined field.

Upload the Schema

Once you have your schema defined in a JSON object, you can upload it to the Knowledge library so that Copilot Editor can reference that file:

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Worker Groups**, select a **Worker Group**.
2. Navigate to **Processing**, then **Knowledge**. Select **Schemas** from the side menu.
3. Select **Add Schema**. In the **New Schema** modal, configure the following settings:
    - **ID:** Enter a unique name for the schema.
    - **Description:** Optionally, add a short description of the schema's contents or purpose.
    - **Schema:** Use the built-in editor to paste the JSON file into the window.



Add a schema to the Knowledge library

4. Select **Save**. Cribl Edge checks your input to ensure it is valid JSON and flags any errors you need to correct before attempting to save again. Select the error icon for more information about the errors that need to be fixed.

5. Confirm that your schema file appears in the list of schemas. Then,commit and deploy the changes to your Fleets.



Consider using Cribl Copilot to help you write commit messages. It analyzes the diff and recommends a message that describes the update.

Build a Pipeline to Transform Your Data Schema

Once you have uploaded your schema to the Knowledge library, you can then reference it when building a Pipeline with Copilot Editor:

1. Check that you have enabled Cribl AI and then launch Copilot Editor.
2. Follow the chat prompts to capture or import sample data. The chat window may also prompt you to select a data source.
3. When the chat window prompts you to choose an objective, select **Convert my input events to a specific target schema**.
4. To use your custom schema, select **Pick from our library**. Use the dropdown menu to select **Custom**, then select your custom schema name from the list. Select **Confirm**.



Copilot Editor limits the size of custom schemas to 100,000 characters. If your file exceeds this

14.6. Manage Metrics And High Cardinality

Cribl Edge provides a streamlined workflow for managing metrics and metric events, especially to reduce high-cardinality data (datasets with a large number of uncommon or unique values).

As you build your Pipeline, you can:

- **Preview and analyze your metrics in Data Preview:** When you add sample data in your Pipeline, the Data Preview pane automatically switches to Metrics View, a metrics-first view designed specifically for viewing and analyzing metrics data. It provides summary statistics and visualizations to help you identify high-cardinality data and similar inefficiencies. This enables you to make informed decisions about data processing before forwarding it to downstream receivers, which can potentially reduce storage costs and improve query performance.
- **Automatically build a Pipeline to process metrics:** As you inspect metrics in Metrics View, you can select the specific metrics or dimensions you want to aggregate or drop. The interface automatically adds Functions to your Pipeline with an initial configuration based on your selection.

This workflow covers a variety of use cases for managing metrics:

- **Reduce high-cardinality data:** Consolidate, aggregate, or drop metrics or dimensions with a large number of unique or uncommon values to reduce time series, simplify data, and improve query performance.
- **Optimize metrics:** Group and segment similar metrics into fewer categories to reduce complexity. Create time-based rollups by aggregating high-frequency metrics into larger intervals (such as hourly or daily). Combine related or duplicate metrics together to create a unified metric.
- **Normalize metrics:** Aggregate metrics collected from different systems into a single, unified view for easier analysis and consistent reporting.

Overview of Metrics Management Workflow

The workflow for using Cribl Edge to optimize metrics has three phases:

1. Add metrics to your Pipeline: Before you start analyzing and managing metrics data, you first need to get metrics data flowing into Cribl Edge. Then, you need to capture or import some sample data to validate your Pipeline as you build it.
2. Analyze metrics in Data Preview: With sample metrics data open in Metrics View in Data Preview, you can analyze the data to find metrics and dimensions that are possibly contributing to issues with high cardinality or large data volumes.
3. Manage metrics: Use the interface to drop and aggregate metrics or drop dimensions. Cribl Edge automates most of the configuration process. Select the metrics you want to aggregate or drop, and Cribl

Edge will automatically add Functions to your Pipeline with an initial configuration based on your selection.

The following sections explain these three phases in more detail.

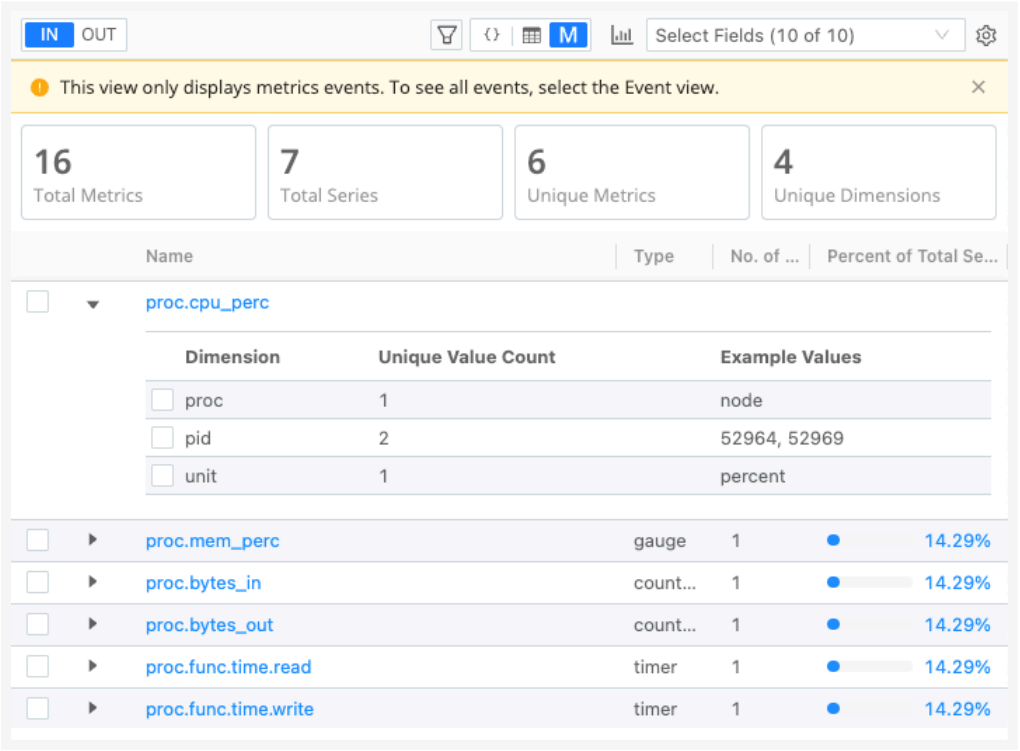
Add Metrics to Your Pipeline

Before you start analyzing and managing metrics data in Cribl Edge, you first need to get metrics data flowing:

1. Add a Source and Destination that can send and receive metrics data. See Integrations for instructions.
2. Add a new Pipeline or open an existing Pipeline. See Pipelines for instructions.
3. Add sample data or capture sample data that contains metrics. See Data Preview: Add Sample Data for instructions.

> **How Data Preview Works**
>
> Pipelines show a list of Functions and their configurations on the left and the Data Preview on the right. The Pipeline processes a set of sample metrics and passes them through Functions in the Pipeline. Then, the Pipeline displays the inbound and outbound results in the Data Preview pane. Whenever you modify, add, or remove a Function, the output in the Data Preview pane updates instantly to reflect your changes. See Data Preview for more information.

4. Because your data sample contains metric events, the Data Preview automatically switches to the Metrics view. It intelligently detects the underlying data structure and switches to this view to display relevant insights.



If your metrics data doesn't show up in Metrics View, it may not be formatted as a supported metric. See Supported Metric Formats and Types for more information.

Analyze Metrics in Data Preview

With sample metrics data open in Metrics View in the Data Preview, you can analyze the data for insights about your metrics. This section explains a variety of strategies for using the Data Preview to find metrics and dimensions that are possibly contributing to downstream storage or processing costs. It also suggests possible solutions and how to implement those solutions in your Pipeline.

Increased downstream storage and processing costs typically have a few common causes:

- **High-cardinality dimensions:** Each combination of dimension values creates a unique time series. If your metrics data has a high number of dimensions with a high number of possible unique values, that means your data has a problem with high cardinality. It has too many possible unique combinations and values (time series). If a downstream vendor charges you by the number of time series in your data, your costs increase with each unique combination of dimensions and dimension values.
- **High-frequency metrics:** If an upstream sender is capturing data points too frequently, you can end up producing a high volume of data. If a downstream vendor charges you by the amount of data you send it, high volumes of metrics data can increase costs or slow down data querying.
- **Low-value metrics:** Not all metrics you produce might be meaningful or valuable to you. Dropping unnecessary metrics or dimensions can help reduce the volume of metrics data so that you only consume what you truly need downstream.

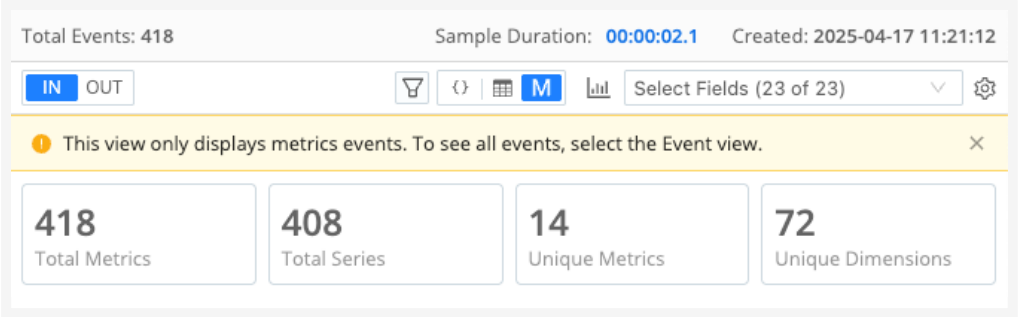
You can use the Metrics View to help detect these common problems in your metrics data.

Get an Overview of Your Metrics

The Data Preview has an **IN** and **OUT** tab so that you compare your incoming data with the changes to the outgoing data. It helps to test and validate the Functions you add to your Pipeline to verify that the data changes applied by the Pipeline match your expectations.

When you first look at the Metrics View, the charts at the top provide high-level summary statistics to help you understand the structure of your metrics, including the total metrics, total time series, the number of unique metrics, and the number of unique dimensions. See Metrics View Reference for detailed descriptions of the Metric View charts and elements.

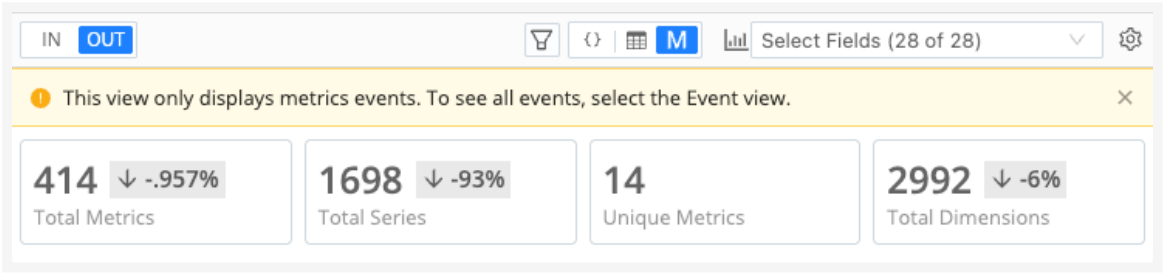
This screenshot shows an example of the summary statistics:



Example of inbound summary statistics in Metrics View

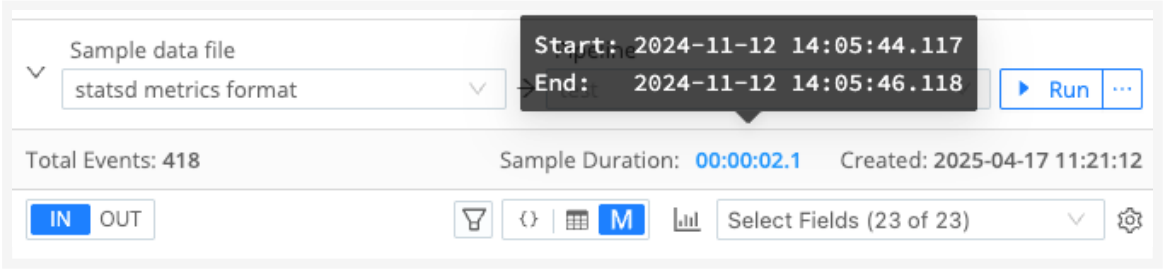
These high-level summaries are useful for before-and-after comparisons of your metrics. After you add Functions that drop or aggregate metrics or dimensions, these summaries display the differences.

This screenshot shows the same summary statistics on the **OUT** tab after applying metrics optimizations in the Pipeline:



Example of outbound summary statistics in Metrics View

To help you evaluate the scope and accuracy of your sample metrics data, the Metrics View in Data Preview also provides a header bar that summarizes key details about the sample. This header bar includes details about the total events, creation date, and the duration of the sampled data. Hovering over the duration reveals the sample's start and end times.



Hover over the Sample Duration to view start and end times

Inspect Time Series

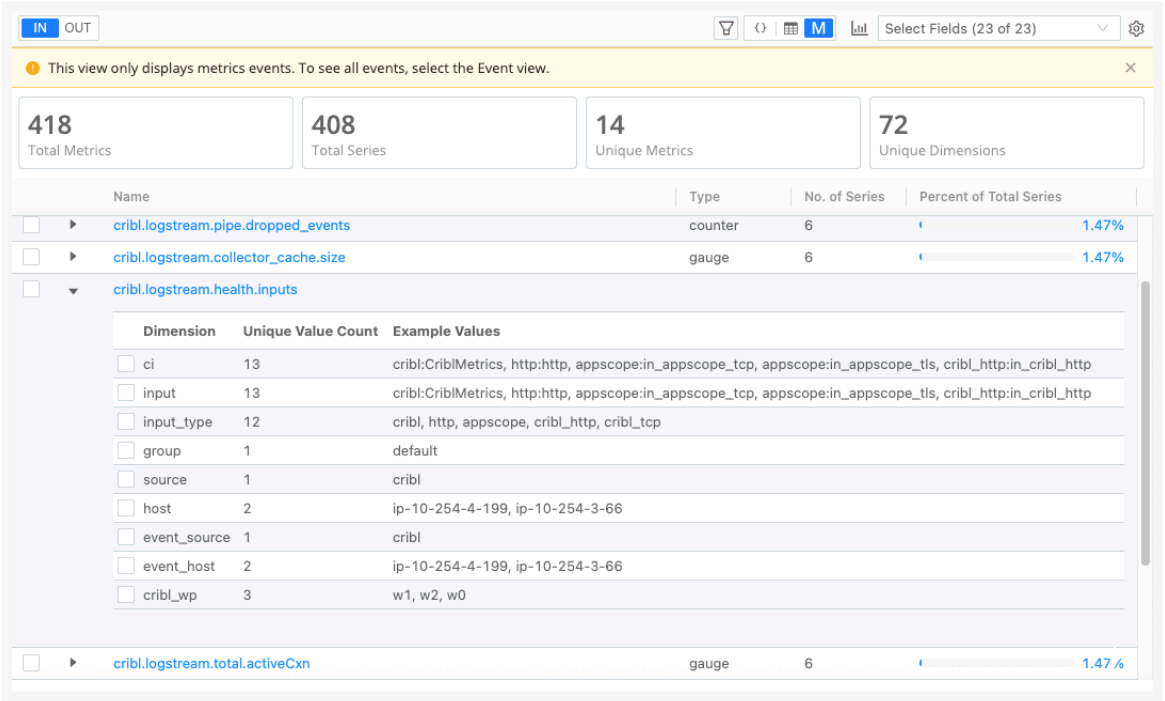
When you first look at the Metrics View, it displays a table listing each metric by name. In that initial view, pay attention to some of these key statistics and indicators for time series:

- **No. of Series:** This column lists the number of time series that this metric and its dimensions creates, calculated as the unique combination of a metric and a dimension pair. If a metric seems to have a high

number listed in this column, that could mean it is contributing to high cardinality.

- **Percent of Total Series**: This column visually represents the proportion of time series represented by this metric, shown as a percentage of the total series. If a metric contributes a higher percentage to the overall total of time series, that could mean it is contributing to high cardinality.

For example, the following screenshot shows a list of metric names. In the list, the metric named `cribl.logstream.health.inputs` stands out because it has a high number of time series (24336) compared to the other metrics. It also represents a significant percentage of total series (97.27%). This metric is likely causing a high-cardinality problem.



Example of a high-cardinality metric in Metrics View

Inspect Dimension Details

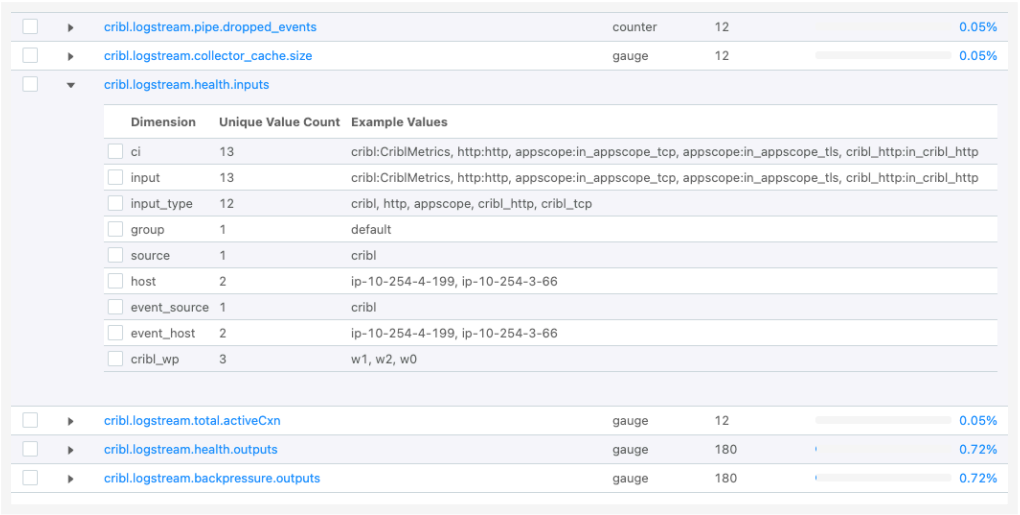
When you want to explore a metric in more detail, you can select the arrow next to the metric name to view its dimensions and the unique values for those dimensions. As you look at the dimension details, examine a few things:

- Are there dimensions with a high number of unique values?
- Do any of dimensions seem to be redundant or share duplicate values?
- As you compare the dimensions, do any seem similar enough that they could be combined together into a broader category or segment?
- Do any of the dimensions seem unnecessary or provide data that isn't valuable for your purposes?

If you answer "yes" to any of these questions, those dimension might be a good candidate for taking action on them in your Pipeline.

To continue the previous example, in the detailed dimensions view for the metric named `cribl.logstream.health.inputs`, several dimensions appear to contain duplicate values:

- `ci` and `input`
- `host` and `event_host`
- `source` and `event_source`



Example of high-cardinality dimensions in Metrics View

Identify Possible Solutions

When you identify metrics or dimensions that contribute to high cardinality and/or high volume, or which don't produce a lot of value for your purposes, you can:

- Drop Metrics: Drop unnecessary metrics to reduce data volume.
- Aggregate Metrics: Aggregate metrics to retain long-term trends while reducing the volume of detailed data, such as keeping averages or summaries only.
- Drop Dimensions: Drop unnecessary or redundant dimensions to minimize the number of unique dimension combinations and the number of time series.
- Aggregate dimensions, such as grouping them into broader categories or rolling them up into higher time frequencies.

Fortunately, Cribl Edge can build these operations into your Pipeline directly from the Metrics View interface, as explained in the next section.

Manage Metrics

Cribl Edge automates most of the configuration process for dropping and aggregating metrics. From the Data Preview, you can select the metric you want to aggregate, and Cribl Edge will automatically add Functions to your Pipeline with an initial configuration based on your selection.

Follow these best practices to troubleshoot common issues and ensure successful configuration when using the Metrics View interface:

- **Troubleshoot metrics data not appearing in Metrics View:** If your metrics data doesn't show up in Metrics View, your date may not be in a supported metric format. See Supported Metric Formats and Types for more information.
- **Find missing check boxes in Data Preview:** If check boxes don't appear next to your metrics data, you may not have added or opened a Pipeline yet. You must add or open a Pipeline to see check boxes.
- **Confirm configuration settings:** Always verify that the configuration settings are correct for your use case before saving.
- **Add and manage multiple Functions:** The interface adds a new Function every time you select metrics to drop or aggregate. Over time, this may lead to a large number of Functions in your Pipeline. To consolidate, you can delete the Functions, then re-add the metrics using the interface. You can also manually combine Functions together by copying and pasting configurations.

Drop Metrics

The Metrics View interface automatically adds a Drop Function to your Pipeline. It pre-configures the **Filter** field with the names of the metrics you select in Data Preview.

To drop metrics from inside the Data Preview:

1. Select the check box next to the names of one or more metrics you want to drop.
2. In the tray that appears under the metrics, select **Drop Metrics**.
3. Check the configuration settings for the Drop Function that now appear in your Pipeline. Optionally, add a **Description** or adjust the configuration settings as needed and select **Save**.
4. Test that you configured the Function correctly by comparing sample incoming data with outgoing data in the Data Preview pane and Pipeline Diagnostics tool. See Data Preview and Metrics View for more information.

Aggregate Metrics

The Metrics View interface automatically adds an Aggregate Metrics Function to your Pipeline. It pre-configures the configuration settings for the metrics you select in Data Preview.

To aggregate metrics from inside the Data Preview:

1. Select the check box next to the names of one or more metrics you want to drop.
2. In the tray that appears under the metrics, select **Aggregate Metrics**.
3. Check the configuration settings for the Aggregate Metrics Function that now appear in your Pipeline and select **Save**. Settings to check or adjust:

- **Description:** Review the generated description and adjust it to better reflect your use case.
- **Final:** Toggle on if this Function is the last Function in your Pipeline.
- **Time window:** Ensure the default duration of the tumbling window for aggregating events meets your needs.
- **Aggregates:** Confirm that the generated aggregation functions match your use case.
- **Group by dimensions:** Check the list of dimensions to group by for accuracy.
- **Output Settings:** Expand this section to reveal additional settings:
  - **Output prefix**: If needed, use a prefix to modify the metric name, making it easier to identify and search for aggregated metrics. A prefix helps distinguish aggregated metrics from non-aggregated ones, allowing you to route or search for them more easily without increasing cardinality.



Check the Aggregate Metrics Function for more information about these settings and additional information about how this Function works.

4. Test that you configured the Function correctly by comparing sample incoming data with outgoing data in the Data Preview pane and Pipeline Diagnostics tool. See Data Preview and Metrics View for more information.

Drop Dimensions

The Metrics View interface automatically adds a Drop Dimensions Function to your Pipeline. It pre-configures the **Filter** field with the names of the dimensions you select in Data Preview. It also adds an initial description to the **Description** field based on your select.

To drop dimensions from inside the Data Preview:

1. Expand a metric to view its dimensions.
2. Select the check box next to the names of one or more dimensions you want to drop.
3. In the tray that appears under the metrics, select **Drop Dimensions**.
4. Check the configuration settings for the Drop Dimensions Function that now appear in your Pipeline. Optionally, edit the **Description** or adjust the configuration settings as needed and select **Save**.
5. Test that you configured the Function correctly by comparing sample incoming data with outgoing data in the Data Preview pane and Pipeline Diagnostics tool. See Data Preview and Metrics View for more information.

Supported Metric Formats and Types

Cribl Edge supports the following metric formats:

- Open Telemetry (OTLP)
- Prometheus
- StatsD

For Open Telemetry metrics, Cribl Edge does not ingest batches. To ingest data from batches, you must enable the **Extract Metrics** setting on your Open Telemetry Source. See OpenTelemetry (OTel) Source for more information.

When Cribl Edge ingests Open Telemetry metrics, it explodes the batches into individual metric events for processing. Cribl Edge can then create new batches from the processed metric data before sending to a downstream receiver.

Metrics View Reference

The Metrics view provides the following high-level summary statistics to help you understand the structure of your metrics event data:

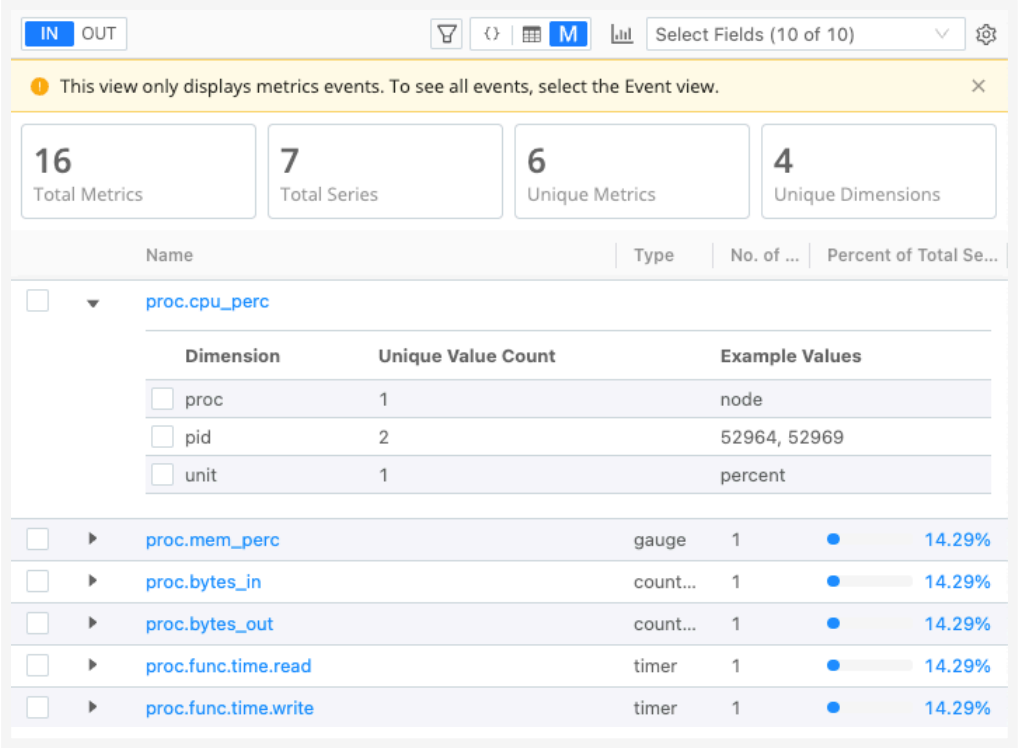
- **Total Metrics**: The total number of metrics in the sample data. If a metric event contains multiple metrics, each one is counted separately.
- **Total Series**: The total number of time series in the metric event, where a time series represents a unique combination of a metric and a dimension pair.
- **Unique Metrics**: The total number of distinct metrics within the sample data.
- **Unique Dimensions**: The total number of dimensions across all metric events.

Beneath the summary, the Metrics view presents a table with the following columns:

- **Name**: The name of a unique metric.
- **Type**: The type of metric, such as a gauge, histogram, and more.
- **No. of Series**: The number of time series that this metric and its dimensions creates, calculated as the unique combination of a metric and a dimension pair.
- **Percent of Total Series**: The proportion of time series represented by this metric, shown as a percentage of the total series.

Each row represents a single metric, which you can expand to show its associated dimensions. The expanded view displays additional details:

- **Dimension**: The name of the dimension.
- **Unique Value Count**: The number of unique values for this dimension, also known as dimensionality.
- **Example Values**: The first five recorded values in the sample data, providing insight into typical values.



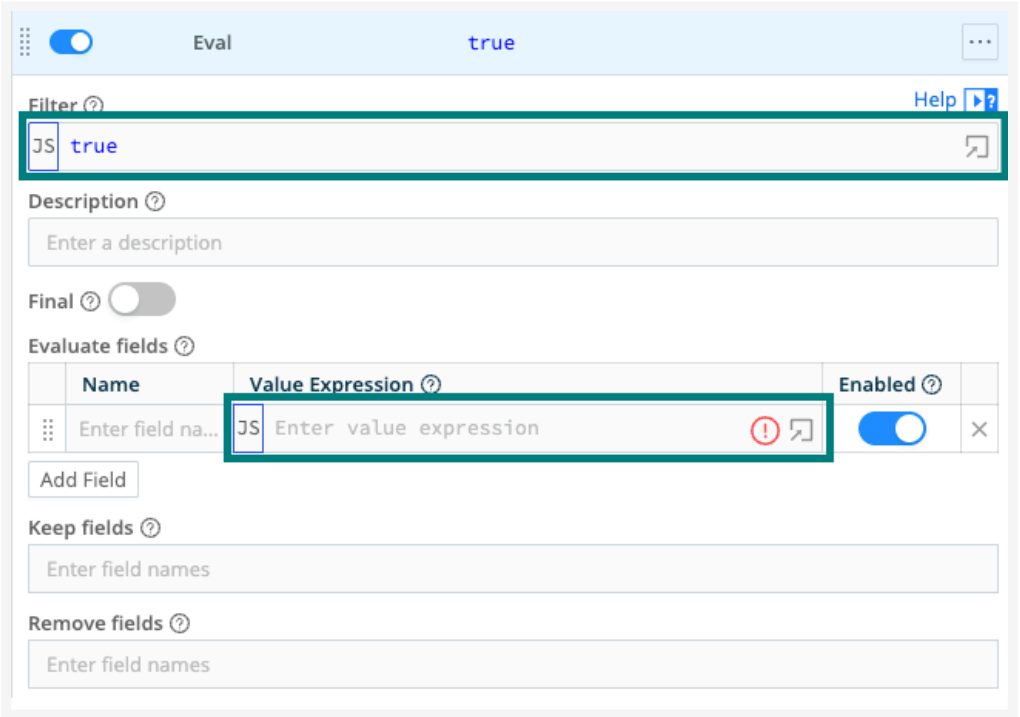
Example Metrics View

14.7. Build Custom Logic To Route And Process Your Data

Throughout Cribl Edge, many settings let you use custom JavaScript code and Cribl Expressions to control how data gets routed or processed. You will encounter JavaScript-enabled fields like these in several parts of Cribl Edge:

- **Routes:** Build custom logic to dynamically control where data gets routed.
- **Functions:**
  - Every Function has a **Filter** field that accepts JavaScript to determine which data the Function will process.
  - Some Functions also include additional fields for applying custom logic to event data.

This screenshot shows an example of two common types of JavaScript expression fields in the Eval Function: a filter expression and a value expression. These fields have a `JS` badge to visually indicate that they accept JavaScript input:



Examples of JavaScript Expression fields

You can use JavaScript syntax in any field that allows JavaScript expressions. You can also use Cribl Expressions, which are pre-built helpers designed for common operations in Cribl Edge. Cribl Expressions begin with `C.`, such as `C.Lookup` or `C.Time`.

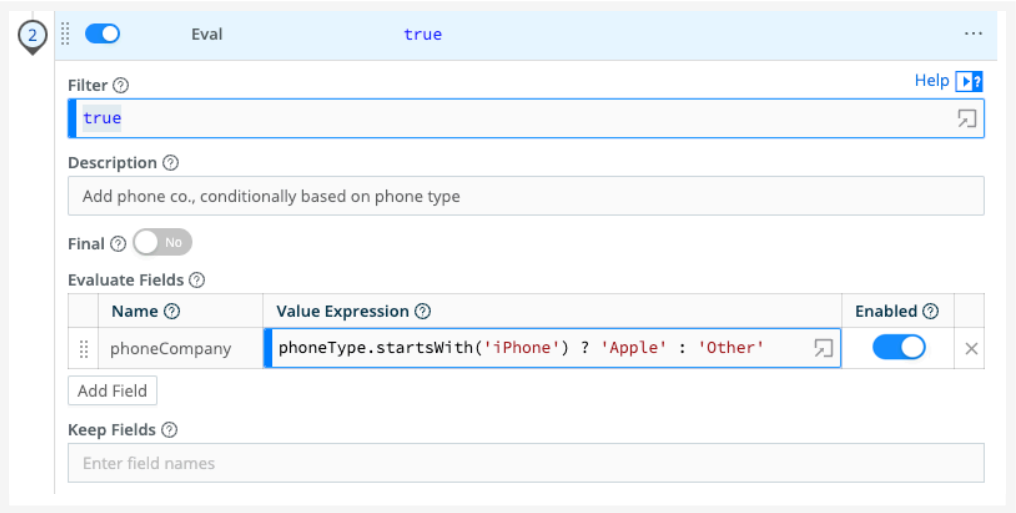
The JavaScript Expressions Sandbox tutorial provides hands-on experience if you would like to learn

more about them through personal experimentation and exploration.

Example Use Case

You can use JavaScript Expression fields to create custom logic for handling data based on your use case. To give a specific use case as an example, perhaps you want to enrich your incoming data by adding a new field called `phoneCompany`, which will assign a value of `Apple` or `Other` based on the contents of another field named `phoneType`.

You could use the Eval Function, which has a **Value Expression** field that accepts custom JavaScript expressions. In that field, you could use the expression `phoneType.startsWith('iPhone') ? 'Apple' : 'Other'`. This JavaScript expression checks if the string `phoneType` starts with `iPhone`. If it evaluates to true, it sets the return value to `Apple`. If not, it sets it to `Other`.



Adding a field to enrich data

Types of JavaScript Fields

Filter Expressions

You will use filter expressions in:

- Routes to dynamically control which data goes to which Pipelines.
- Functions to set the scope or narrow down which data gets processed by that Function.

The **Filter** field on Functions or Routes can accept custom filter expressions to control whether a Function or Route processes each event. Filter expressions must evaluate to either `true` or `false`:

- If the result is `true`, the data gets processed by that Route or Function.

- If the result is `false`, the data will pass through unchanged.

By default, the **Filter** field is set to `true`, meaning the Route or Function applies to all events. You can replace this with a custom JavaScript expression to apply the Route or Function only to data that matches specific conditions.

Value Expressions

Use value expressions in Functions to assign or calculate values dynamically. Use these expressions to create new fields, modify existing ones, or perform transformations on data.

For example, you can:

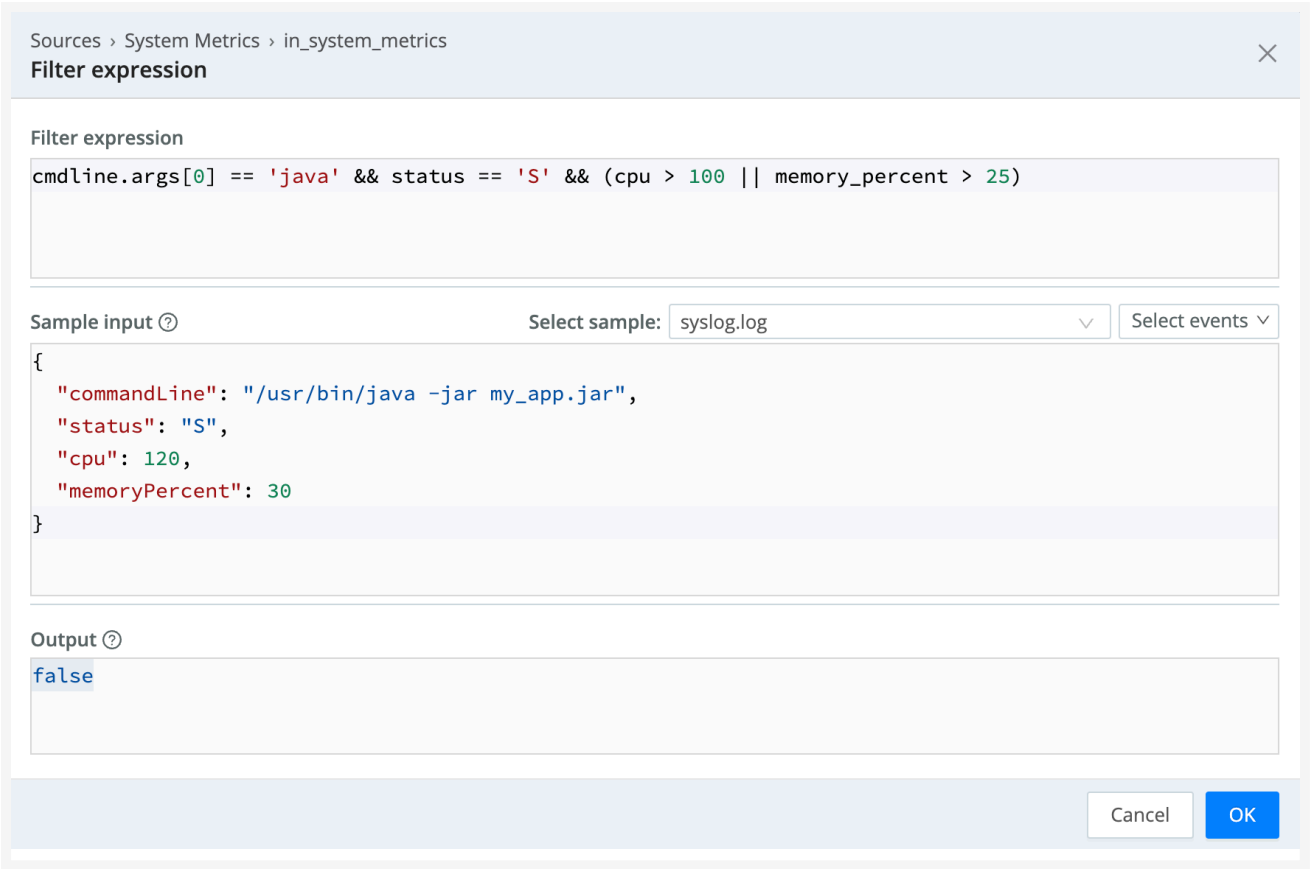
- Convert a timestamp to an hour-based value: `Math.floor(_time / 3600)`
- Mask part of a Source string: `source.replace(/.{3}/, 'XXX')`

Anywhere a Function lets you define a value with JavaScript, you can use a value expression to apply custom logic based on the content of the event.

Test and Validate Expressions in Advanced Mode

To test your filter expression against sample input data, click the **Advanced mode** button within the **Filter expression** field. The **Filter expression** modal will open.

Here, you can paste in your JavaScript expression to filter the data. You can also select a sample JSON input from the drop-down or enter your own sample data directly in the **Sample input** field.



The **Advanced mode** window

The **Output** shows you whether your expression is valid or not:

- For filter expressions, the output will resolve to `true` if it is a valid expression, which means that it can correctly apply the custom filter to the sample data. A value of `false` means you need to fix the syntax of the expression or ensure it applies to the sample data.
- For value expressions, the output will show you if the filter expression returns any matching processes, based on the filter expressions and JSON input.

In most cases, you can use Advanced Mode to test and validate JavaScript that uses Cribl Expressions as well.

> If you are testing an expression that uses the Lookup Cribl Expression (`C.Lookup`), be aware of lookup limitations. To avoid overloading your web browser, lookup files larger than 10 MB are not loaded and will not evaluate in Advanced Mode. However, if the expression is valid, the updated values are still correctly applied to route and process data.

Best Practices to Create Predictable Expressions

- In a value expression, ensure that the source variable is not `null`, `undefined`, or `empty`. For example, assume you want to have a field called `len`, to be assigned the length of a second field called

`employeeID`. But you're not sure if `employeeID` exists. Instead of `employeeID.length`, you can use a safer shorthand, such as: `(employeeID || '').length`.

- If a field does not exist (undefined), and you're doing a comparison with its properties, then the boolean expression will **always** evaluate to false. For example, if `employeeID` is undefined, then both of these expressions will evaluate to false: `employeeID.length > 10` , and `employeeID.length < 10` .
- `==` means "equal to," while `===` means "equal value **and** equal type." For example, `5 == 5` and `5 == "5"` each evaluate to **true**, while `5 === "5"` evaluates to **false**.
- A ternary operator is a very powerful way to create conditional values. For example, if you wanted to assign either `minor` or `adult` to a field `groupAge`, based on the value of `age`, you could do: `(age >= 18) ? 'adult' : 'minor'`.

Fields with Non-Alphanumeric Characters

If there are fields whose names include non-alphanumeric characters, such as `@timestamp` or `user-agent` or `kubernetes.namespace_name`, you can access them using `__e['<field-name-here>']`. (Note the single quotes.) More details are available on the Mozilla Developer Docs topic on Property accessors..

In any other places that reference the field, such as the Eval function's field names, you should use a single-quoted literal, of the form: `'<field-name-here>'`.

Wildcard Lists

Wildcard lists appear in various Functions, such as Eval, Mask, Publish Metrics, Parser, and Serialize.

Wildcard lists accept strings with asterisks (`*`) to represent one or more terms. They also accept strings that start with an exclamation mark (`!`) to **negate** one or more terms. This allows for implementing any combination of allowlists and blocklists.

Wildcard fields (`*`) support two use cases:

- Wildcards (`*`) to include multiple fields.
- Nested addressing, where you can reference a parent field and all its children using a wildcard. For example, if you convert `_raw` into an object, use `_raw*` to refer to `_raw` and all nested fields inside it.

Wildcard lists are order-sensitive, evaluated from left to right. This is especially relevant when you use negated terms. For negations to take precedence over wildcards when evaluated, you must list negations before wildcards.

Some examples:

15. Functions

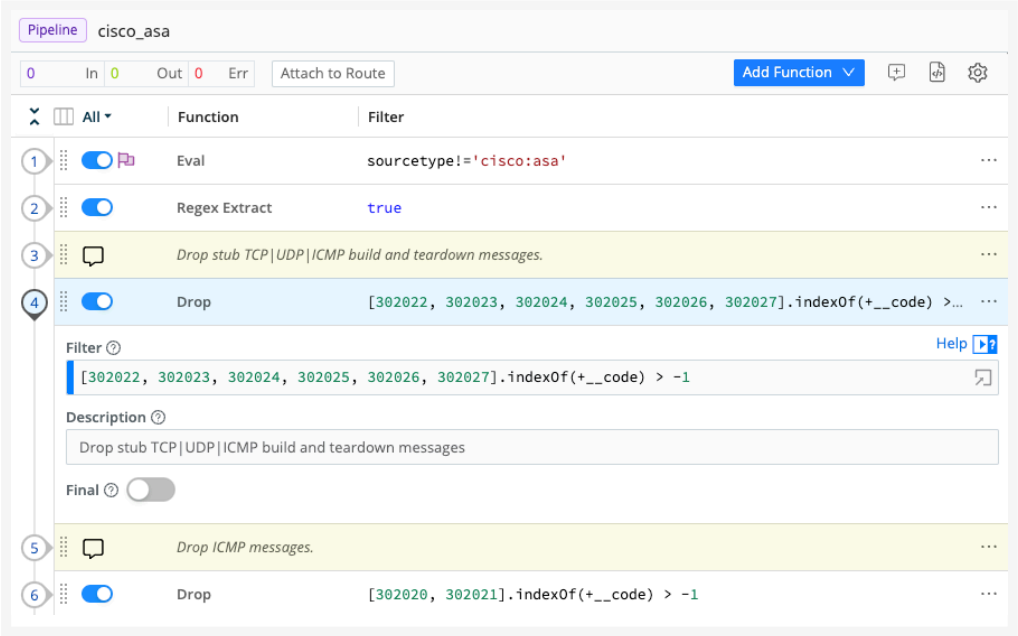
When events enter a Pipeline, they're processed by a series of Functions. At its core, a Function is code that executes on an event, and it encapsulates the smallest amount of processing that can happen to that event.

The term "processing" means a variety of possible options: string replacement, obfuscation, encryption, event-to-metrics conversions, and so on. For example, a Pipeline can be composed of several Functions - one that replaces the term `foo` with `bar`, another one that hashes `bar`, and a final one that adds a field (say, `dc=jfk-42`) to any event that matches `source=='us-nyc-application.log'`.

How Do They Work

Functions are atomic pieces of JavaScript code that are invoked on each event that passes through them. To help improve performance, configure Functions with filters to further scope their invocation to matching events only. See Build Custom Logic to Route and Process Your Data: Filter Expressions for more information.

You can add as many Functions in a Pipeline as necessary, though the more you have, the longer it will take each event to pass through. Also, you can enable and disable Functions within a Pipeline as necessary. This lets you preserve structure as you optimize or debug.



Functions stack in a Pipeline

You can reposition Functions up or down the Pipeline stack to adjust their execution order. Use a Function's left grab handle to drag and drop it into place.

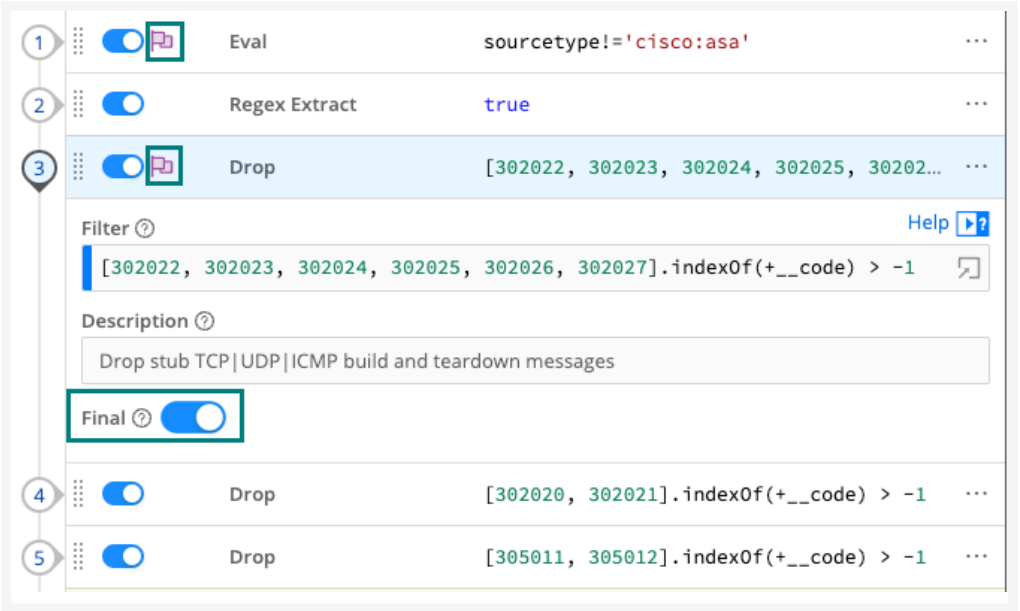
The Final Toggle

The **Final** toggle in Function settings controls what happens to the results of a Function.

When **Final** is toggled off (default), events will be processed by this Function, **and** then passed on to the next Function below.

When **Final** is toggled on, the Function "consumes" the results, meaning they will not pass down to any other Function below.

A flag in Pipeline view indicates that a Function is **Final**.



The Eval and first Drop Functions bearing the Final flag

Functions and Shared-Nothing Architecture

Cribl Edge is built on a shared-nothing architecture, where each Node and its **Worker Processes operate separately, and process events independently of each other**. This means that all Functions operate strictly in a Worker Process context - state is not shared across processes.

For example, consider two events that meet a Pipeline's criteria to be aggregated. If these two events arrive on **separate** Workers or Worker Processes, Cribl Edge will **not** aggregate them together.

This is particularly important to understand for certain Functions that might imply state-sharing, such as Aggregations, Rollup Metrics, Dynamic Sampling, Sampling, Suppress, and so on.

If you have a large number of Worker Processes, consider implementing a distributed caching tier, such as Redis, to aggregate events across Workers. (See our Redis Function topic.)

Out-of-the-Box Functions

Cribl Edge ships with several Functions out-of-the-box, and you can chain them together to meet your requirements. For more details, see individual **Functions**, and the **Use Cases** section, within this documentation.

Accessing Event Fields with `__e`

The special variable `__e` represents the `(context)` event inside a JavaScript expression. Using `__e` with square bracket notation, you can access any field within the event object, for example, `__e['hostname']`. Make sure to use single quotes around the field name. Functions use `__e` extensively.

You also **must** use this notation for fields that contain a special (non-alphanumeric) character like `user-agent`, `kubernetes.namespace_name`, or `@timestamp`. For more details, see MDN's Property Accessors documentation. In any other place where such fields are referenced - for example, in an Eval Function's Field names - you should use a single-quoted literal, of the form: `'<field-name-here>'`.

Supported JavaScript Standard

Cribl Edge supports the ECMAScript® 2015 Language Specification.

Regex Flavor

Regular expressions in Cribl Edge use the JavaScript regex engine, which conforms to ECMAScript 2015. This applies to all regex-related Functions, including Regex Extract, Regex Filter, Mask, and others, as well as Event Breakers and the Regex Library.

If you test patterns in external tools such as Regex101, select the **ECMAScript (JavaScript)** flavor to ensure compatibility.

Very Large Integer Values

Cribl Edge's JavaScript implementation can safely represent integers only up to the Number.MAX_SAFE_INTEGER constant of about 9 quadrillion (precisely, {2^53}-1). Cribl Edge Functions will round down any integer larger than this, in Data Preview and other contexts. Trailing 0's might indicate such rounding down of large integers.

Specify Fields' Precedence Order in Expressions

For Sources that support adding **Fields (Metadata)**, you can use a **Value** expression to allow fields in events to override the predefined field values. For example, the following expression's left-to-right OR logic ensures that if an inbound event includes an `index` field, its value is used: `${__e['index'] || 'myIndex'}`.

If no `index` field is present, the expression falls back to the constant `myIndex` defined in the expression.

What Functions to Use When

- Add, remove, update fields: Eval, Lookup, Regex Extract
- Find & Replace, including basic `sed`-like, obfuscate, redact, hash, and so on: Mask, Eval
- Add GeoIP information to events: GeoIP
- Extract fields: Regex Extract, Parser
- Extract timestamps: Auto Timestamp
- Drop events: Drop, Regex Filter, Sampling, Suppress, Dynamic Sampling
- Sample events (such as high-volume, low-value data): Sampling, Dynamic Sampling
- Suppress events (such as duplicates): Suppress
- Serialize events to CEF format (send to various SIEMs): CEF Serializer
- Serialize / change format (for example, convert JSON to CSV): Serialize
- Serialize compliant events into SNMP traps: SNMP Trap Serialize
- Convert JSON arrays into their own events: JSON Unroll, XML Unroll
- Flatten nested structures (for example, nested JSON): Flatten
- Aggregate events in real-time (statistical aggregations): Aggregations
- Convert events to metrics format: Publish Metrics, Prometheus Publisher (beta)
- Transforms dimensional metrics events into the OTLP (OpenTelemetry Protocol) format: OTLP Metrics
- Resolve hostname from IP address: Reverse DNS (beta)
- Resolve account names from Windows SID (**Cribl Edge only**): Windows SID Lookup
- Extract numeric values from event fields, converting them to type `number`: Numerify
- Send events out to a command or a local file, via `stdin`, from any point in a Pipeline: Tee
- Convert an XML event's elements into individual events: XML Unroll
- Duplicate events in the same Pipeline, with optional added fields: Clone
- Break events **within**, instead of before they reach, a Pipeline: Event Breaker
- Add a text comment within a Pipeline's UI, to label steps without changing event data: Comment

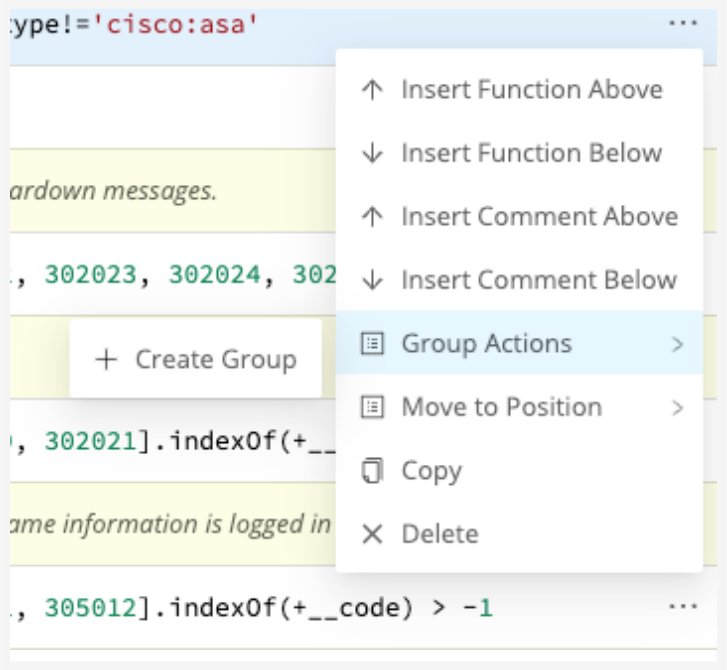
For more usage examples, download the official Tip Sheet: Cribl Stream.

Function Groups

A Function group is a collection of consecutive Functions that can be moved up and down a Pipeline's Functions stack together. Groups help you manage long stacks of Functions by streamlining their display. They are a UI visualization only, purely for visual context, and do not affect the movement of data through Functions. While Functions are in a group, those Functions maintain their global position order in the Pipeline: data moves down the listed Functions, ignoring any grouping assignments.

Function groups work much like Route groups.

To build a group from any Function, select the Function's ••• (Options) menu, then select **Group Actions** > **Create Group**.

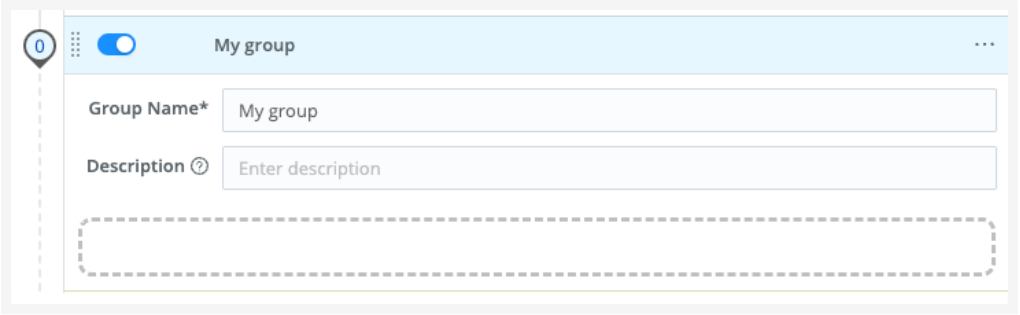


Creating a group

You'll need to enter a **Group Name** before you can save or resave the Pipeline. Optionally, enter a **Description**.

Once you've saved at least one group to a Pipeline, other Functions' ••• (Options) > **Group Actions** submenus will add options to **Move to Group** or **Ungroup/Ungroup All**.

You can also use a Function's left grab handle to drag it into, or out of, a group. A saved group that's empty displays a dashed target into which you can drag Functions.


0
My group
Group Name*
My group
Description
Enter description

15.1. Aggregations

The Aggregations Function computes summary statistics (like averages, sums, and counts) from event data.



Each Worker Process executes this Function independently on its share of events. For details, see Functions and Shared-Nothing Architecture.

Usage

Filter: Filter expression (JavaScript) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Simple description about this Function. Defaults to empty.

Final: Toggle on to stop feeding data to the downstream Functions. Default is toggled off.

Time window: The time span of the tumbling window for aggregating events. Must be a valid time string (e.g., `10s`). Must match pattern `\d+[sm]$`.

Aggregates: Aggregation function(s) to perform on events. E.g., `sum(bytes).where(action=='REJECT').as(TotalBytes)`. Expression format: `aggFunction(<FieldExpression>).where(<FilterExpression>).as(<outputField>)`. See more examples below.



When used without `as()`, the aggregate's output will be placed in a field labeled `<fieldName>_<aggFunction>`. If a conflict occurs, the last aggregate takes precedence. For example, given two aggregates - `sum(bytes).where(action=='REJECT')` and `sum(bytes)` - the latter one (`bytes_sum`) takes precedence.

Group by fields: Fields to group aggregates by. Supports wildcard expressions.



Using the wildcard character `*` causes causes all fields in the event to be included in the aggregation, which can result in high cardinality and increased memory usage. To avoid this impact, first exclude fields that can result in high cardinality, **then** introduce wildcards, if any. For example: `!_time, !_numericValue, *`

Evaluate fields: Set of key-value pairs to evaluate and add/set. Fields are added in the context of an aggregated event, before they're sent out. Does not apply to passthrough events.

Time Window Settings

Cumulative aggregations: Toggle on to retain aggregations for cumulative aggregations when flushing out an aggregation table event. Toggle off (default) to reset aggregations to `0` on flush.

Lag tolerance: The lag tolerance represents the tumbling window tolerance to late events. Must be a valid time string (e.g., `10s`). Must match pattern `\d+[sm]$`.

Idle bucket time limit: The amount of time to wait before flushing a bucket that has not received events. Must be a valid time string (e.g., `10s`). Must match pattern `\d+[sm]$`.

Output Settings

Passthrough mode: Toggle on to pass through the original events along with the aggregation events. Default is toggled off.

Metrics mode: Determines whether to output aggregates as metrics. The default is off, which outputs aggregates as events. Toggle on if you are working with metrics-heavy data.



If you toggle on **Metrics mode**, consider also toggling on **Treat dots as literals**, which is under **Advanced Settings**. This setting is strongly recommended if your metric names or dimensions use dot notation, as used in StatsD and Open Telemetry metrics formats.

Sufficient stats mode: Determines whether to output *only* statistics sufficient for the supplied aggregations. Default is toggled off, meaning output richer statistics.

Preserve group by fields: Determines whether to preserve the structure of the original aggregation event's group by fields.

Output prefix: A prefix that is prepended to all of the fields output by this Aggregations Function.

Advanced Settings

Aggregation event limit: The maximum number of events to include in any given aggregation event. Defaults to unlimited. Must be at least `1`.

Aggregation memory limit: The maximum memory usage for aggregations within each Worker Process. This limit is applied per process, not globally. By default, it's unlimited, allowing the use of all available system memory. Accepts numerals with multiple-byte units like KB, MB, GB (for example, `4GB`).

Treat dots as literals: Toggle on if your fields or dimensions use dot notation, which are used in StatsD and Open Telemetry metrics formats. Examples: `os.name` or `os.version`. If toggled off (default), metrics or fields

using dot notation are enclosed in quotation marks and treated as strings. Example outputs when toggled off: `"os.name"` or `"os.version"`. Use the Data Preview to confirm that the data output matches your expectations.

Flush on stream close: Toggle on (default) if you want aggregations to flush when an input stream is closed. If toggled off, the Time Window Settings will control flush behavior; this can be preferable in cases like the following:

- Your input data consists of many small files.
- You are sending data to Prometheus. Enabling **Flush on stream close** can send Prometheus multiple aggregations from the same Worker Process for the same time period. Prometheus cannot tell the multiple aggregations apart, and will ingest only the first one.

Aggregation Functions

- `avg(expr:FieldExpression)`: Returns the average `expr` values.
- `count(expr:FieldExpression)`: If `expr` is omitted, returns the number of events received over the time window. If `expr` is given, returns the number of events seen where `expr` evaluates to a value other than null or undefined. For example, with these three events in the same time window:
  - `{ _time: 20, _val: 5 }`
  - `{ _time: 22 }`
  - `{ _time: 24, _val: null }`

  `count()` would return `3`, while `count(_val)` would return `1`.
- `dc(expr: FieldExpression, errorRate: number = 0.01)`: Returns the estimated number of distinct values of `expr`, within a relative error rate. Lower error rates increase the accuracy of the result, at the cost of using more memory. For example, with these three events in the same time window:
  - `{ _time: 20, _name: 'Alice' }`
  - `{ _time: 22, _name: 'Bob' }`
  - `{ _time: 24, _name: 'Alice' }`

  `dc(_name)` would return `2`.
- `distinct_count(expr: FieldExpression, errorRate: number = 0.01)`: Identical to `dc(expr)`.
- `earliest(expr:FieldExpression)`: Returns the earliest (based on `_time`) observed `expr` value.
- `first(expr:FieldExpression)`: Returns the first observed `expr` value.
- `histogram(expr:FieldExpression, buckets: number[])`. Returns the average of the values of `expr` and generates a field with the same name as the aggregated output field, suffixed with `_data`.
  - `buckets` must contain at least one numeric value.

The `_data` field in the output contains three pieces of information:

- `_sum` - the sum of all values of `expr`.
- `_count` - the number of events seen where `expr` evaluates to a value other than null or undefined.
- `_buckets` - an object whose keys are the buckets defined in the function, and whose values are the number of values that fall in that histogram bucket. In addition to the buckets defined by the user, the object will include a bucket labeled `Infinity`, into which all values will be placed. A value of `x` falls into a histogram bucket if it is less than or equal to the value of the bucket. A value of `45` would fall into a bucket with value `50`, but not one with value `40`. Counts are cumulative; all values counted in a bucket will also be counted in every bucket larger than it. For example, with these three events in the same time window:
  - `{ _time: 20, age: 20 }`
  - `{ _time: 22, age: 25 }`
  - `{ _time: 24, age: 30 }`

  `histogram(age, [10, 25, 40])` would result in the following on the output event:

```
{
  age_histogram: 25
  age_histogram_data: {
    _count: 3,
    _sum: 75
    _buckets: {
      10: 0,
      25: 2,
      40: 3,
      Infinity: 3
    }
  }
}
```

Copy

- `last(expr:FieldExpression)`: Returns the last observed `expr` value.
- `latest(expr:FieldExpression)`: Returns the latest (based on `_time`) observed `expr` value.
- `list(expr:FieldExpression[, max:number, excludeNulls: boolean = true])`: Returns a list of the observed values of `expr`.
  - Optional `max` parameter limits the number of values returned. If omitted, the default is `100`. If set to `0`, will return all values.

    - Optional `excludeNulls` boolean excludes null and undefined values from results. If included, defaults to `true`.
- `max(expr:FieldExpression)`: Returns the maximum `expr` value.
- `median(expr:FieldExpression)`: Returns the middle value of the sorted parameter.
- `min(expr:FieldExpression)`: Returns the minimum `expr` value.
- `mode(expr: FieldExpression[, excludeNulls: boolean = true])`: Returns the single most frequently encountered `expr` value.
    - Optional `excludeNulls` boolean excludes null and undefined values from results. If included, defaults to `true`.
- `per_second(expr:FieldExpression)`: Returns the rate of change of the values of `expr` over the aggregate time window. This is equivalent to `rate(expr, '1s')`. For example, with these three events in the same time window:
    - `{ _time: 20, _val: 5 }`
    - `{ _time: 22, _val: 15 }`
    - `{ _time: 24, _val: 35 }`

  `per_second(_val)` would give back `(35 - 5) / (24 - 20) = 7.5`, meaning that `_val` increased by 7.5 every second over the time window.
- `perc(level: number, expr: FieldExpression)`: Returns <level> percentile value of the numeric `expr` values.
- `rate(expr:FieldExpression, timeString: string = '1s')`: Returns the rate of change of the values of `expr` over the aggregate time window. Calculated as (latest value - earliest value) / (latest timestamp - earliest timestamp) * number of seconds in `timeString`. For example, with these three events in the same time window:
    - `{ _time: 20, _val: 5 }`
    - `{ _time: 22, _val: 15 }`
    - `{ _time: 24, _val: 35 }`

  `rate(_val, '2s')` would give back `(35 - 5) / (24 - 20) * 2 = 15`, meaning that `_val` increased by 15 every 2 seconds over the time window.
- `stdev(expr:FieldExpression)`: Returns the sample standard deviation of the `expr` values.
- `stdevp(expr:FieldExpression)`: Returns the population standard deviation of the `expr` values.
- `sum(expr:FieldExpression)`: Returns the sum of the `expr` values.
- `summary(expr:FieldExpression)[, quantiles: number[]])`: Returns the average of the `expr` values and generates a field with the same name as the aggregate output field, suffixed with `_data`.
    - Optional: `quantiles` values must be between `0` and `1` inclusive.

  The `_data` field in the output contains three pieces of information:

- `_sum` - the sum of all `expr` values.
- `_count` - the number of events seen where `expr` evaluates to a value other than null or undefined.
- `_quantiles` - an object whose keys are the quantiles defined in the function, and whose values are the quantile value across the `expr` values. For example, a quantile key `0.5` and value `500` would indicate that the 50th percentile (or median) of the values seen was 500.
- For example, with these three events in the same time window:
  - `{ foo: 100 }`
  - `{ foo: 200 }`
  - `{ foo: 300 }`

  `summary(foo, [0.25, 0.5, 0.75])` would result in the following on the output event:

```
{
  foo_summary: 25
  foo_summary_data: {
    _count: 3,
    _sum: 75
    _quantiles: {
      0.25: 100,
      0.5: 150,
      0.75: 225
    }
  }
}
```

Copy

- `sumsq(expr:FieldExpression)`: Returns the sum of squares of the `expr` values.
- `top(expr: FieldExpression, count: number[, excludeNulls: boolean = true])`: Returns the most frequently encountered `expr` values, up to `count` number of results.
  - `count` parameter must be a positive integer.
  - Optional `excludeNulls` boolean excludes null and undefined values from results. If included, defaults to `true`.
- `values(expr:FieldExpression[, max: number, excludeNulls: boolean = true, errorRate: number])`: Returns a list of distinct `expr` values.
  - Optional `max` parameter limits the number of values returned; if omitted, the default is `0`, meaning return all distinct values.

    - Optional `excludeNulls` boolean excludes null and undefined values from results. If included, defaults to `true`.
    - Optional `errorRate` parameter controls how accurately the function counts "distinct" values. Range is `0-1`; if omitted, the default value is `0.01`. Higher values allow higher error rates (fewer unique values recognized), with the offsetting benefit of less memory usage.
- `variance(expr:FieldExpression)`: Returns the sample variance of the `expr` values.
- `variancep(expr:FieldExpression)`: Returns the population variance of the `expr` values.

Safeguarding Data

Upon shutdown, Cribl Edge will attempt to flush the buffers that hold aggregated data, to avoid data loss. If you set a **Time window** greater than 1 hour, Cribl recommends adjusting the **Aggregation memory limit** and/or **Aggregation event limit** to prevent the system from running out of memory.

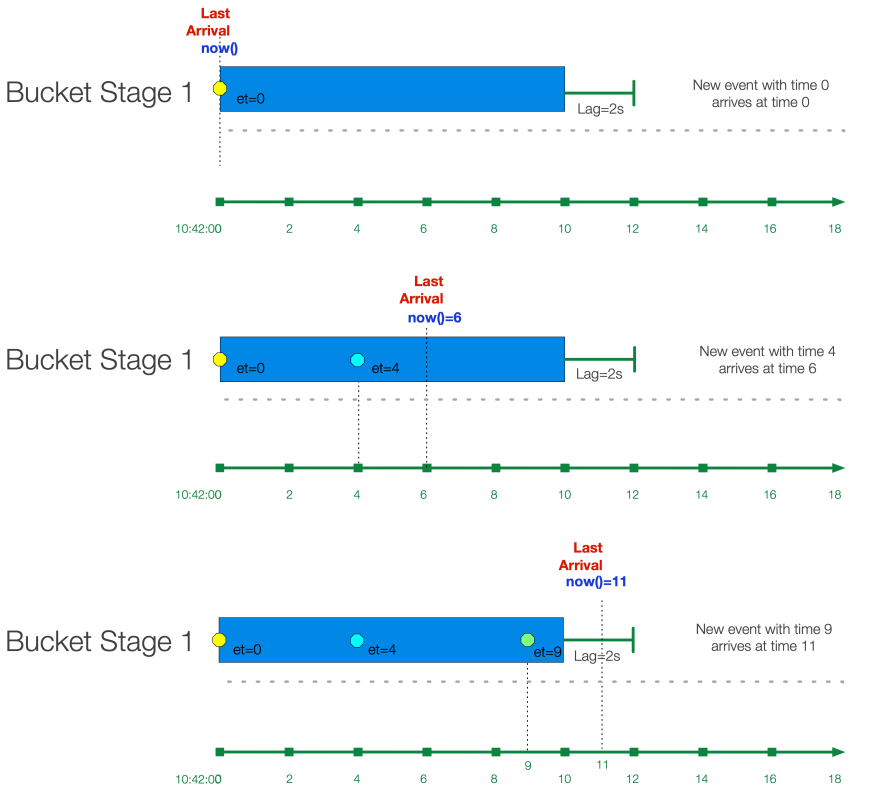
This is especially necessary for high-cardinality data. (Both settings default to unlimited, but we recommend setting defined limits based on testing.)

How Do Time Window Settings Work?

Lag Tolerance

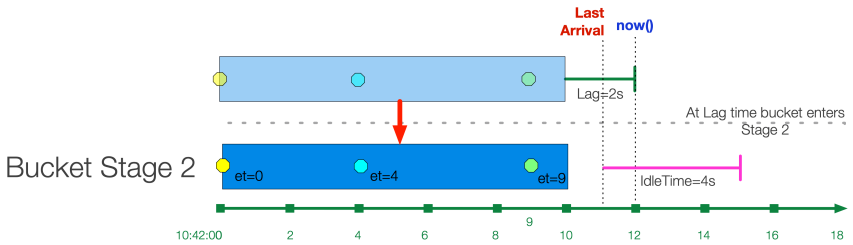
As events are aggregated into windows, there is a good chance that most will arrive later than their event time. For instance, given a `10s` window (`10:42:00 – 10:42:10`), an event with timestamp `10:42:03` might come in 2 seconds later at `10:42:05`.

In several cases, there will also be late, or lagging, events that will arrive **after** the latest time window boundary. For example, an event with timestamp `10:42:04` might arrive at `10:42:12`. Lag Tolerance is the setting that governs how long to wait - after the latest window boundary - and still accept late events.



The “bucket” of events is said to be in Stage 1, where it’s still accepting new events, but it’s not yet finalized. Notice how in the third case, an event with event time `10:42:09` arrives 1 second past the window boundary at `10:42:11`, but it’s still accepted because it happens before the lag time expires.

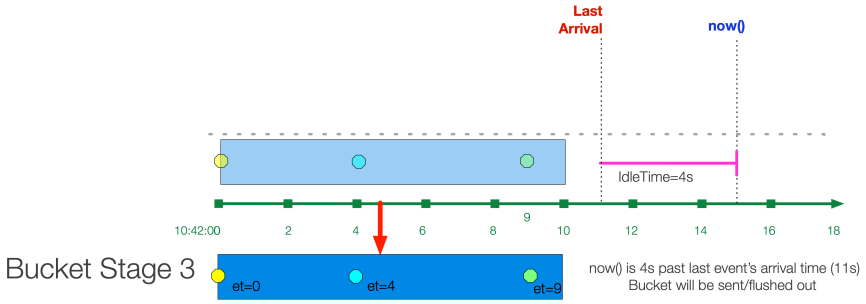
After the lag time expires, the bucket moves to Stage 2.



If the bucket is created from a historic stream, then the bucket is initiated in Stage 2. Lag time is not considered. A “historic” stream is one where the latest time of a bucket is before `now()`. E.g., if the window size is 10s, and `now()=10:42:42`, an event with `event_time=10` will be placed in a Stage 2 bucket with range `10:42:10 - 10:42:20`.

Idle Bucket Time Limit

While Lag Tolerance works with event time, Idle Bucket Time Limit works on arrival time (i.e., real time). It is defined as the amount of time to wait before flushing a bucket that has not received events.



After the Idle Time limit is reached, the bucket is “flushed” and sent out of the system.

Examples

Assume we’re working with VPC Flowlog events that have the following structure:

```
version account_id interface_id srcaddr dstaddr srcport dstport protocol packets bytes start end action log_status
```

For example:

```
2 99999XXXXX eni-02f03c2880e4aaa3 10.0.1.70 10.0.1.11 9999 63030 6 6556 262256 1554562460 1554562475 ACCEPT OK 2 496698360409 eni-08e66c4525538d10b 37.23.15.38 10.0.2.232 4373 8108 6 1 52 1554562456 1554562466 REJECT OK
```

Scenario A:

Every 10s, compute sum of `bytes` and output it in a field called `TotalBytes`.

Time Window: `10s` Aggregations: `sum(bytes).as(TotalBytes)`

Scenario B:

Every 10s, compute sum of `bytes`, output it in a field called `TotalBytes`, group by `srcaddr`.

Time Window: `10s` Aggregations: `sum(bytes).as(TotalBytes)` Group by fields: `srcaddr`

Scenario C:

Every 10s, compute sum of `bytes` but only where action is `REJECT`, output it in a field called `TotalBytes`, group by `srcaddr`.

Time Window: `10s` Aggregations: `sum(bytes).where(action=='REJECT').as(TotalBytes)` Group by Fields: `srcaddr`

Scenario D:

Every 10s, compute sum of `bytes` but only where action is `REJECT`, output it in a field called `TotalBytes`. Also, compute distinct count of `srcaddr`.

15.2. Aggregate Metrics

The Aggregate Metrics Function computes aggregate statistics for metrics and metric events. It can perform common aggregations such as count, sum, average, min, max, median, rate, and more. This Function helps you pre-aggregate your metrics data before sending it to downstream storage or analysis tools to reduce storage needs and costs. It also simplifies large datasets by reducing complexity and improving query efficiency.

Use this Function to:

- **Reduce high-cardinality data:** Consolidate datasets with a large number of unique or uncommon values to improve query performance and save money on storage or processing costs.
- **Create time-based rollups:** Aggregate high-frequency metrics into larger time intervals (such as hourly or daily) to simplify analysis and improve long-term data retention.
- **Normalize metrics from multiple sources:** Aggregate metrics collected from different systems into a single, unified view.
- **Combine related or duplicate metrics:** Merge related metrics together to create a unified metric.
- **Downsample metrics:** Retain long-term trends while reducing the volume of detailed data.

How the Aggregate Metrics Function Works

The Aggregate Metrics Function processes data in the following order:

1. **Filter:** Checks incoming data for metrics or metric events that match the specified filter values.
2. **Aggregate:** Combines the applicable metrics or metric events based on the configuration settings.
3. **Output:** Extracts, formats, and outputs the aggregated values.

When using this Function, keep these considerations in mind:

- The default metric type is `Automatic`, which allows the aggregation function to determine the appropriate output type.
- Not all methods for aggregating metrics (count, sum, average, and so forth) can convert aggregate data into a certain metrics types. For example, histograms can only convert to other histograms. See Metric Types for available types, recommendations, and supported aggregation functions.



The Aggregate Metrics Function handles metrics data only. In this context, this refers to events with the internal field `__criblMetrics`. If you need to aggregate other types of data, such as logs and traces, use the Aggregations Function instead. If the Aggregate Metrics Function receives non-metric data that do not contain the `__criblMetrics` internal field, it will pass through unchanged without any effect.

Configure the Aggregate Metrics Function

To simplify configuration, the Pipeline Data Preview pane can automate most of the configuration process for the Aggregate Metrics Function. When you build or edit a Pipeline with metrics data, the Data Preview pane automatically detects metrics in your sample data and switches to the Metrics View, which organizes metrics by name. From this view, you can select the metrics you want to aggregate, and Cribl Edge will automatically add an Aggregate Metrics Function to your Pipeline with an initial configuration based on your selection. See Manage Metrics and High Cardinality for more information.

To configure this Function manually:

1. Add a Source and Destination to begin generating metrics. See Integrations for more information.
2. Add a new Pipeline or open an existing Pipeline. Add or capture sample data test your Pipeline as you build it. See Pipelines for more information.
3. At the top of the Pipeline, select **Add Function** and search for `Aggregate Metrics`, then select it.
4. In the **Aggregate Metrics** modal, configure the following general settings:
    - **Filter:** A JavaScript filter expression that selects which metrics to process through the Function. Defaults to `true`, meaning it evaluates all events.
        - If you don't want to use the default setting `true`, you can use a JavaScript expression to apply this Function to specific metric events. See Usage Examples: Filter Expressions for examples.
        - If you use the Metrics View interface in the Pipeline Data Preview to add this Function to your Pipeline, Cribl Edge automatically builds a filter expression based on the metrics you selected. See Manage Metrics and High Cardinality for more information.
    - **Description:** A brief description of how this Function modifies your metrics to help other users understand its purpose later. Defaults to empty.
        - If you use the Metrics View interface in the Pipeline Data Preview to add this Function to your Pipeline, Cribl Edge automatically generates a description based on the selected metrics, including usage tips. Review the generated description and adjust it to better reflect your use case. See Manage Metrics and High Cardinality for more information.
    - **Final:** When enabled, stops data from continuing to downstream Functions for additional processing. Default: Off. Toggle **Final** on if Aggregate Metrics is the last Function in your Pipeline.
    - **Time window:** The duration of the tumbling window for aggregating events. Must be a valid time string (such as `10s`) and match the pattern `\d+[sm]$`. See Time Window Settings for additional settings.
    - **Aggregates**: A table that defines how to aggregate specific metric events. If you use the Metrics View interface in Pipeline Data Preview to add this Function to your Pipeline, Cribl Edge

automatically populates this field. See Manage Metrics and High Cardinality for more information. The table includes two inputs:

- **Metric type**: Specifies the type of metric output to apply. Defaults to `Automatic`. Be aware not all aggregation functions can convert aggregate data into a certain metrics types. For example, histograms can only convert to other histograms. See Metric Types for available types, recommendations, and supported functions.
- **Aggregration:** Enter a JavaScript expression that performs the desired aggregation function (such as count, average, sum, and more). See Usage Examples: Aggregation Function Expressions for a full list of available aggregation functions and usage examples.

- **Group by dimensions**: Defines one or more dimensions to group aggregates by. List excluded dimensions first, followed by the dimensions to group by. This field is not a JavaScript expression. Instead it uses a lightweight syntax that accepts simple strings and operators, including wildcard characters. Accepted formats:

Syntax	Description
`!string`	Exclude the specified dimension or field from the grouping. Example: `!proc`.
`string`	Include the specified dimension or field in the grouping. Example: `proc`.
`!*`	Exclude all dimensions or fields from the grouping. Wildcard characters are not supported for dimensions or fields with dotted field names, such as StatsD and Open Telemetry formats.
`*`	Include all dimensions or fields in the grouping. **NOTE:** Use with caution. The wildcard character cannot search or filter for any dimensions or fields that use dot notation, such as `dimension.name` or `field.name`. StatsD and Open Telemetry use dot notation formats.
`string*` or `"string*"`	Include all dimensions or fields that begin with this string pattern. **NOTE:** To search or filter for dimensions or fields that use dot notation, enclose the string in quotation marks. Example: `"os.name"` or `"os.version"`. StatsD and Open Telemetry use dot notation formats.



Using the wildcard character `*` includes all dimensions in the aggregation, which can increase cardinality and lead to higher memory usage. To avoid this impact, first exclude dimensions that may cause high cardinality, then introduce wildcards, if needed. For example: `!_time, !_numericValue, *`

- **Evaluate fields**: A set of key-value pairs to evaluate and add or update. The system adds fields in the context of an aggregated event before sending them out. This setting does not apply to passthrough events.

5. Configure additional settings as needed. For more information, see:
    - Time Window Settings
    - Output Settings
    - Advanced Settings
6. Test that you configured the Function correctly by comparing sample incoming data with outgoing data in the Data Preview pane and Pipeline Diagnostics tool. See Data Preview and Metrics View for more information.

Time Window Settings

Cumulative aggregations: When enabled, retains cumulative aggregation values when flushing an aggregation table event. When disabled (default), resets aggregation values to `0` on flush.

Lag tolerance: Specifies the tolerance for late events in a tumbling window. Must be a valid time string (such as `10s`) and match the pattern `\d+[sm]$`.

Idle bucket time limit: The amount of time to wait before flushing a bucket that has not received events. Must be a valid time string (such as `10s`) and match the pattern `\d+[sm]$`.

Output Settings

Passthrough mode: When enabled, passes the original events along with the aggregation events. Default: Off.

Sufficient stats mode: Controls whether to output *only* the statistics required for the supplied aggregations. When disabled (default), outputs richer statistics.

Preserve group by fields: Controls whether to retain the original aggregation event's group by field structure.

Output prefix: Adds an optional prefix to all fields output by the Aggregate Metrics Function. Use a prefix to modify the metric name, making it easier to identify and search for aggregated metrics. A prefix helps distinguish aggregated metrics from non-aggregated ones, allowing you to route or search for them more easily without increasing cardinality.

Advanced Settings

Aggregation event limit: The maximum number of events to include in a single aggregation event. Defaults to unlimited. Minimum value: `1`.

Aggregation memory limit: The maximum memory that aggregations can use. Defaults to unlimited (the amount of memory available in the system). Accepts numerals with units such as KB, MB, and GB. Example: `4GB`.

Treat dots as literals: Toggle on (default) if your fields or dimensions use dot notation, which are used in StatsD and Open Telemetry metrics formats. Examples: `os.name` or `os.version`. If toggled off, metrics or fields using dot notation are enclosed in quotation marks and treated as strings. Example outputs when toggled off: `"os.name"` or `"os.version"`. Use the Data Preview to confirm that the data output matches your expectations.

Flush on stream close: When enabled (default), flushes aggregations when an input stream closes. When disabled, the Time Window Settings control flush behavior. Disabling this setting may be preferable in cases such as:

- Your input data consists of many small files.
- You are sending data to Prometheus. Enabling Flush on stream close can cause Prometheus to receive multiple aggregations from the same Worker Process for the same time period. Prometheus cannot distinguish between them and will ingest only the first one.

Metric Types

This table explains the differences between the metric types and which aggregration functions they support:

Type	Description
Automatic	Default. Select this type to allow the aggregation function determine the appropriate metric output type.
Counter	A counter is a cumulative metric that increases over time. It represents a value that only increments or resets to zero. Counters never decrease unless the system restarts or the counter is manually reset. Use this metric type to monitor changes over time where you need to compute the rate of increase. If you need to capture the current value at the time of measurement rather than a cumulative value, use the gauge metric type instead. Supports aggregation for these functions: • count (recommended for the counter metric type) • avg • distinct_count • dc • earliest • latest • max • max • min • median • mode • perc

Type	Description
	• per_second • rate • stdev • stdev • stdevp • sum • sumsq • summary • variance • variancep
Distribution	**NOTE:** Available for Datadog only. A distribution is a collection of related measurements that capture the spread or variation of data points over time. Instead of tracking a single value, it records a set of values, allowing you to calculate statistics like mean, median, percentiles, and standard deviation. Use this metric type to identify patterns and outliers in performance. Supports aggregation for these functions: • avg • count • distinct_count • dc • earliest • latest • first • histogram • last • max • min • median • mode • perc • per_second • rate • stdev • stdevp • sum

Type	Description
	• sumsq • summary • variance • variancep
Gauge	A gauge represents a single value at a specific point in time. Unlike a counter, a gauge can increase or decrease to reflect the current state rather than its cumulative value. Use this metric type to provide a snapshot of a value at a particular moment. Supports aggregation for these functions: • sum (recommended for the gauge metric type) • avg • count • distinct_count • dc • earliest • latest • first • histogram • last • max • median • min • mode • per_second • perc • rate • stdev • stdevp • sumsq • summary • variance • variancep
Histogram	A histogram measures the distribution of a set of values over time. It tracks both the count of events and how those events are distributed across a range of values. Use this metric type to measure the frequency of values falling within predefined ranges. Histograms can be useful for identifying outliers and performance bottlenecks.

Type	Description
	Supports aggregation for histograms only.
Summary	A summary is similar to a histogram but computes specific quantiles over time (such as median or 90th percentile)instead of counting values in predefined ranges. Use this metric type to capture statistical trends in specific statistical measures over time. Supports aggregation for summary only.
Timer	A timer measures the duration of an event from start to finish. It records how long an operation takes and allows you to calculate statistics like min, max, average, and percentiles over time. Use this metric type when when your focus is specifically on measuring elapsed time. Supports aggregation for these functions: • avg • count • distinct_count • dc • earliest • latest • first • histogram • last • max • median • min • mode • perc • per_second • rate • stdev • stdevp • sum • sumsq • summary • variance • variancep

Usage Examples

This section provides typical JavaScript expressions that you might commonly use with the Aggregate Metrics Function.

Filter Expressions

The **Filter** setting is a JavaScript expression that selects which data to process through the Aggregate Metrics Function. The following is an example of a JavaScript expression that filters for three metrics named `proc.cpu_perc`, `proc.mem_perc`, and `proc.bytes_in` and other variants that may include these names:

```
(_metric == 'proc.cpu_perc' || __criblMetrics[0].nameExpr.includes("'proc.cpu_perc'")
(_metric == 'proc.mem_perc' || __criblMetrics[0].nameExpr.includes("'proc.mem_perc'")
(_metric == 'proc.bytes_in' || __criblMetrics[0].nameExpr.includes("'proc.bytes_in'")
```

Copy

Aggregation Function Expressions

The **Aggregates** table in the Aggregate Metric Function defines how to aggregate specific metric events. You can use the **Aggregation** column in this table to enter a JavaScript expression that performs the desired aggregation function. The Aggregate Metrics Function shares much of its functionality with the Aggregations Function. For more detailed examples, see the Aggregations Function documentation.

This table provides examples of JavaScript expressions for common aggregation functions for the Aggregate Metrics Function:

Aggregation	Example
Average	`avg(_value || proc.mem_perc).as(proc.mem_perc_avg)`
Count	`count(_value || proc.bytes_in).as(proc.bytes_in_count)`
Sum	`sum(_value || proc.mem_perc).as(proc.mem_perc_sum)`

Cribl Edge places the output of the aggregate in a field `<fieldName> <aggFunction>`. If a conflict

15.3. Auto Timestamp

The Auto Timestamp Function extracts time to a destination field, given a source field in the event. By default, Auto Timestamp makes a first best effort and populates `_time`. When you add a sample (via paste or a local file), you should accomplish time and event breaking at the same time you add the data.

This Function allows fine-grained and powerful transformations to populate new time fields, or to edit existing time fields. You can use the Function's Additional timestamps section to create custom time fields using regex and custom JavaScript `strptime` functions.



The Auto Timestamp Function uses the same basic algorithm as Event Breakers and the C.Time.timestampFinder() native method.

Usage

Filter: Filter expression (JavaScript) that selects data to feed through the Function. The default `true` setting passes all events through the Function.

Description: Simple description about this Function. Defaults to empty.

Final: Toggle on to stop feeding data to the downstream Functions. Default is toggled off.

Source field: Field to search for a timestamp. Defaults to `_raw`.

Destination field: Field to place extracted timestamp in. Defaults to `_time`. Supports nested addressing.

Default timezone: Select a timezone to assign to timestamps that lack timezone info. Defaults to `Local`. (This drop-down includes support for legacy names: `EST5EDT`, `CST6CDT`, `MST7MDT`, and `PST8PDT`.)

Additional timestamps: To extract additional timestamp formats, select **Add Timestamp** to define each format. Each row will provide these fields:

- **Regex**: Regex, with first capturing group matching the timestamp. You can type in a pattern, or use one of the predefined Regexes.
- **Strptime format**: Select or enter the `strptime` format for the captured timestamp.

Advanced Settings

Time expression: Expression with which to format extracted time. Current time, as a JavaScript Date object, is in global `time`. Defaults to `time.getTime() / 1000`. You can access other fields' values via `__e.`

`<fieldName>`.



For details about Cribl Edge's Library (native) time methods, see: C.Time - Time Functions.

Start scan offset: How far into the string to look for a time string.

Max timestamp scan depth: Maximum string length at which to look for a timestamp.

Default time: How to set the time field if no timestamp is found. Defaults to **Current time**.

Two fields enable you to constrain (clamp) the parsed timestamp, to prevent the Function from mistakenly extracting non-time values as unrealistic timestamps:

- **Earliest timestamp allowed**: Enter a string that specifies the latest allowable timestamp, relative to now. (Sample value: `-42years`. Default value: `-420weeks`.) Parsed values earlier than this date will be set to the **Default time**.
- **Future timestamp allowed**: Enter a string that specifies the latest allowable timestamp, relative to now. (Sample value: `+42days`. Default value: `+1week`.) Parsed values after this date will be set to the **Default time**.

Format Reference

This references https://github.com/d3/d3-time-format#locale_format. Directives annotated with a `(†)` symbol might be affected by the locale definition.

```
%a - abbreviated weekday name. (†)
%A - full weekday name. (†)
%b - abbreviated month name. (†)
%B - full month name. (†)
%c - the locale's date and time, such as %x, %X. (†)
%d - zero-padded day of the month as a decimal number [01,31].
%e - space-padded day of the month as a decimal number [ 1,31]; equivalent to %_d.
%f - microseconds as a decimal number [000000, 999999].
%H - hour (24-hour clock) as a decimal number [00,23].
%I - hour (12-hour clock) as a decimal number [01,12].
%j - day of the year as a decimal number [001,366].
%m - month as a decimal number [01,12].
%M - minute as a decimal number [00,59].
%L - milliseconds as a decimal number [000, 999].
%p - either AM or PM. (†)
%Q - [Unix time](https://en.wikipedia.org/wiki/Unix_time) (milliseconds).
%s - Unix time (seconds).
```

```
%S - second as a decimal number [00,61].
%u - Monday-based (ISO 8601) weekday as a decimal number [1,7].
%U - Sunday-based week of the year as a decimal number [00,53].
%V - ISO 8601 week of the year as a decimal number [01, 53].
%w - Sunday-based weekday as a decimal number [0,6].
%W - Monday-based week of the year as a decimal number [00,53].
%x - the locale's date, such as %-m/%-d/%Y. (†)
%X - the locale's time, such as %-I:%M:%S %p. (†)
%y - year without century as a decimal number [00,99].
%Y - year with century as a decimal number.
%Z - time zone offset, such as -0700, -07:00, -07, or Z.
%% - a literal percent sign (%).
```

Complying with the Format

In order to use auto timestamping upon ingestion, the formatting used must match the `%Z` parameters above. For example, this Function will automatically parse all of these formats:

- `2020/06/10T17:17:35.004-0700`
- `2020/06/10T17:17:35.004-07:00`
- `2020/06/10T17:17:35.004-07`
- `2020/06/10T10:17:35.004Z`
- `2020/06/10T11:17:35.004 EST`

To parse other formats, you can use the Additional Timestamps section's internal **Regex** or **Strptime Format** operators.

Basic Example

Create a Pipeline with one Auto Timestamp Function. Leave the settings of the Function to their defaults.

Sample event:

```
Jan 20 12:03:55 PA-VM 1,2025/01/20 13:03:58,CRIBL,TRAFFIC,end,2049,2024/09/20
14:03:58,314.817.108.226,10.0.0.102,314.817.108.226,10.0.2.65,cribl,,,incomplete,vsys1,u
forwarding-default,2024/09/20
13:03:58,574326,1,53722,8088,53722,8088,0x400064,tcp,allow,296,296,0,4,2024/09/20
13:03:45,7,any,0,730277,0x0,United States,10.0.0.0-10.255.255.255,0,4,0,aged-
out,0,0,0,0,,PA-VM,from-policy,,,0,,0,,N/A,0,0,0,0
```

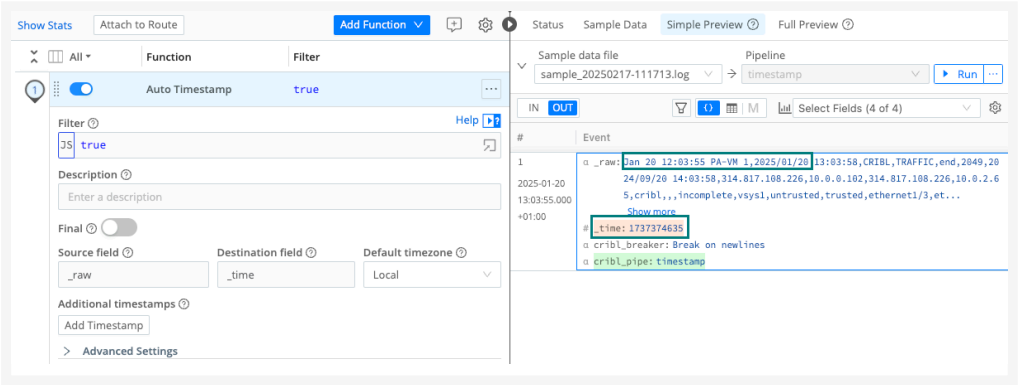
To add this sample (after creating an Auto Timestamp Function with the above **Filter** expression), in the Pipeline page go to **Sample Data** > **Import Data**, and paste the data snippet above. Do not make any changes to timestamping or line breaking, and select **Save as Sample File**.

By default, Cribl Edge will inspect the first 150 characters, and will extract the first valid timestamp it sees. You can modify this character limit under **Advanced Settings** > **Max Timestamp Scan Depth**.

Cribl Edge will grab the first part of the event, and will settle on the first matching value to display for `time`:

- `_time: 1737374635`
- **GMT**: Jan 20, 2025 12:03:55 PM GMT
- **Your Local Time**: Jan 20, 2025 1:03:55 PM GMT+01:00

Because no explicit timezone has been set (under **Default Timezone**), `_time` will inherit the **Local** timezone, which in this example is `GMT +01:00`.



Auto Timestamp Basic Example

Timezone Dependencies and Details

Cribl Edge uses ICU for timezone information. It does not query external files or the operating system. The bundled ICU is updated periodically.

For additional timezone details, see: https://www.iana.org/time-zones.

Advanced Settings Example

The `datetime.strptime()` method creates a datetime object from the string passed in by the **Regex** field.

Here, we'll use `datetime.strptime()` to match a timestamp in AM/PM format at the end of a line.

Sample:

```
This is a sample event that will push the datetime values further on inside the event.
This is still a sample event and finally here is the datetime information!:
```

`Server_UTC_Timestamp="04/27/2020 2:30:15 PM"`

Max timestamp scan depth: `210`

Click to add **Additional timestamps**:

Regex: `(\d{1,2}\/\d{2}\/\d{4}\s\d{1,2}:\d{2}:\d{2}\s\w{2})`

Strptime format: `'%m/%d/%Y %H:%M:%S %p'`

Gnarly Details

- This Function supports the `%f` (microseconds) directive.
- Cribl Edge will truncate timestamps to three-digit (milliseconds) resolution, omitting trailing zeros.
- For further examples, see Extracting Timestamps from Messy Logs.

15.4. CEF Serializer

The CEF Serializer takes a list of fields and/or values, and formats them in the Common Event Format (CEF) standard. CEF defines a syntax for log records. It is composed of a standard prefix, and a variable extension formatted as a series of key-value pairs.

Format

`CEF:Version|Device Vendor|Device Product|Device Version|Device Event Class ID|Name|Severity|[Extension]`

Usage

Filter: Filter expression (JavaScript) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Simple description about this Function. Defaults to empty.

Final: Toggle on to stop feeding data to the downstream Functions. Default is toggled off.

Output field: The field to which the CEF formatted event will be output. Nested addressing supported. Defaults to `_raw`.

Header Fields

CEF Header field definitions. The field values below will be written pipe (|)-delimited in the Output Field. Names cannot be changed. Values can be computed with a JavaScript expression, or can be constants.

- **cef_version**: Defaults to `CEF:0`.
- **device_vendor**: Defaults to `Cribl`.
- **device_product**: Defaults to `Cribl`.
- **device_version**: Defaults to `C.version`.
- **device_event_class_id**: Defaults to `420`.
- **name**: Defaults to `Cribl Event`.
- **severity**: Defaults to `6`.

Extension Fields

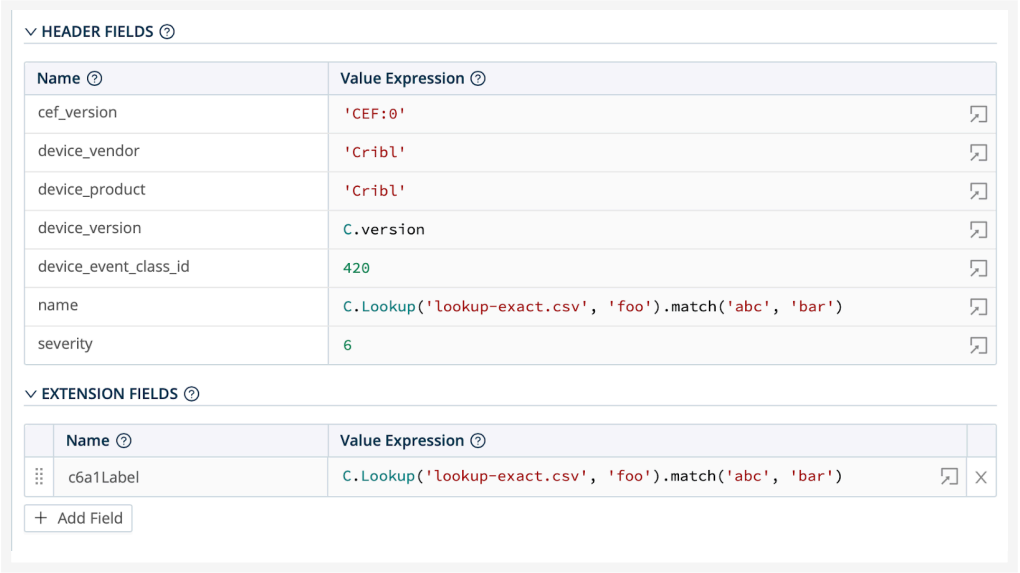
CEF Extension field definitions. Field names and values will be written in `key=value` format. Select each field's Name from the drop-down list. Values can be computed with a JavaScript expression, or can be constants.

Example

For each CEF field, allowed values include strings, plus any custom Cribl function. For example, if using a lookup:

Name: `Name` Value expression: `C.Lookup('lookup-exact.csv', 'foo').match('abc', 'bar')`

This can be used for any of the `CEF` **Header Fields**.



The resulting event has the following structure for an **Output Field** set to `_CEF_out`:

```
_CEF_out:CEF:0|Cribl|Cribl|42.0-61c12259|420|Business Group
6|6|c6a1Label=Colorado_Ext_Bldg7
```

15.5. Chain

The Chain Function does one thing: It connects (or *chains*) data processing from a Pipeline or Pack to another Pipeline or Pack. This can be useful for sequential processing, or just to separate groups of related Functions into discrete Pipeline or Pack units that make intuitive sense.

Control Flow

The chained Pipeline or Pack will, upon completion, normally return control to the parent Pipeline/Pack containing the Chain Function. However, if the chained Pipeline/Pack contains any Function with the **Final** flag enabled, processing will stop there. In this case, back in the parent Pipeline/Pack, no Function below Chain will execute.

Cycle Detection and Throughput

The Chain Function includes guardrails against circular references. In v.4.3 and newer, Cribl Edge detects cycles when you configure your Pipeline. This early detection (compared to older versions' runtime detection) means that:

- If you try to save a Pipeline with a cyclical reference, Cribl Edge will throw an error.
- If an existing Pipeline's configuration contains a cycle, Cribl Edge will disable the final Chain Function involved in the cycle.

Despite these safeguards, Cribl recommends that you keep chained configurations simple and understandable by all your users. Also, keep in mind that inserting a Chain Function can impose a slight performance hit, compared to including all processing in the original Pipeline or Pack.

Pipeline Versus Pack Scope

You will see different scope restrictions when using Chain in a Pipeline versus in a Pack:

- In a Pipeline, the **Processor** drop-down displays both Pipelines and Packs as targets to chain to.
- In a Pack, the **Processor** drop-down offers only Pipelines contained within that Pack.

Usage

Filter: Filter expression (JavaScript) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

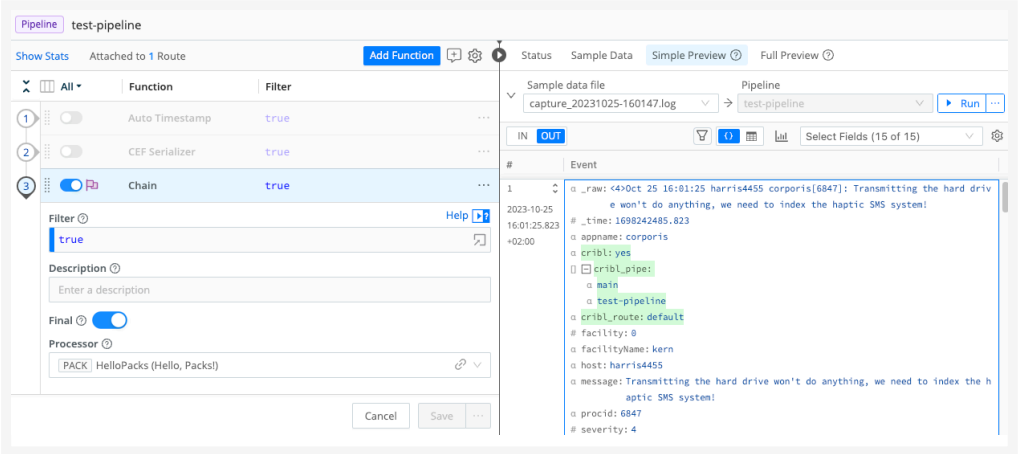
Description: Optionally, add a simple description of this Function's purpose. Defaults to empty.

Final: Toggle on to stop feeding data to the downstream Functions. Default is toggled off. (Note that this will not prevent data from flowing to the Function's defined **Processor**.)

Processor: Use this drop-down to select a configured Pipeline or Pack through which to forward events.

Example

This shows a simple preview of a `pipeline-1` Pipeline, which chains to a `pipeinpipe` Pipeline. Notice that each event's added `cribl_pipe` field lists all Pipelines/Packs through which the event was chained.



cribl_pipe field shows whole processing path

15.6. Clone

The Clone Function creates a copy of events in the same Pipeline, with optional added fields. Cloned events go to the same Destination as the original event, because they are in the same Pipeline. If your Destination is an Output Router, you can use filters to send cloned events to different Destinations.

Usage

Filter: Filter expression (JavaScript) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Simple description about this Function. Defaults to empty.

Final: Toggle on to stop feeding data to the downstream Functions. Default is toggled off.

Clones: Create clones with the specified fields added and set.

Fields: Set of key-value pairs to add. Nested addressing is supported.

Examples

Staging Example

In this example, the Destination will receive a clone with an `env` field set to `staging`.

Field: `env` **Value**: `staging`

Index Example

In this scenario, we insert a Clone Function at the beginning of a Pipeline to create cloned events. We can later use these events as a baseline to compare against the original events, after various Functions have processed them.

You can assign any meaningful fields to the cloned events - anything that will help you identify them when comparing. This example simply assigns a key-value pair of `index: clones`.

Field: `index` **Value**: `clones`

To keep the cloned events from being processed by Functions later in the same Pipeline, you'll need to specify `index!='clones'` in their **Filter** expressions.

15.7. Code

The Code Function provides a way to execute custom JavaScript logic directly within your data processing pipeline. This Function helps you solve complex data transformation challenges that require procedural logic, loops, or conditional handling beyond what the standard Functions provide.

> **For Advanced Users Only**
>
> The Code Function should only be configured by experienced JavaScript developers. Because this Function executes custom scripts, poorly validated code can cause issues such as infinite loops or failed returns, which can significantly increase your throughput volume or degrade system performance.
>
> Use this Function with extreme care. Always test and validate your logic in a development environment before deploying to production.

Cribl Edge's standard Functions, such as Eval, cover the majority of scenarios that users typically need to implement. You should use Code Functions only as a last resort, such as when you need to construct a complex block of code.

Use this Function to:

- **Execute custom or complex procedural logic:** Perform multi-step transformations that require variables, loops, and nested conditional statements.
- **Handle non-standard data structures:** Parse or rebuild deeply nested objects or custom proprietary formats that don't fit into standard schemas or patterns.

How the Code Function Works

The Code Function processes each incoming event in the following order:

1. **Input validation:** The Function identifies the incoming event and ensures it is available for processing within the JavaScript environment.
2. **Script execution:** Cribl Edge executes your custom code against the event. Your logic can create new fields or it can read, modify, or delete existing ones.
3. **Error handling and logging:** If the script encounters an error (such as a syntax issue or a runtime exception), the Function captures the error. It then logs the diagnostic information based on your configured error log sample rate. See Advanced Settings for more information about this setting.

> The Code Function executes your script line-by-line. If an error occurs midway through your code, the Function stops immediately for that event. However, any changes made to the event prior to

> the error are retained. This can result in partially processed events entering your Pipeline. See the example in the Partially Processed Events section of the troubleshooting guide for more information.

4. **Event return:** The Function returns the processed event back to the Pipeline for additional processing or routing.

Prohibited Elements

Generally, any syntax forbidden in JavaScript Strict Mode is also forbidden within the Code Function. To ensure system stability and security, Cribl Edge disallows using the following JavaScript features and objects.

The following JavaScript features and objects are not allowed:

- **Global objects:** `window`, `global`, `globalThis`.
- **Timers and async:** `Promises`, `setTimeout`, `setInterval`.
- **Dynamic execution:** `eval`, `uneval`, `Function (constructor)`.
- **Logging:** `console` (use the `C.log` utility instead).
- **Data Structures:** `set`.

Supported Logic

Cribl Edge supports the ECMAScript 2015 Language Specification. This means you can use standard procedural logic and array methods including loops, array methods, and standard math.

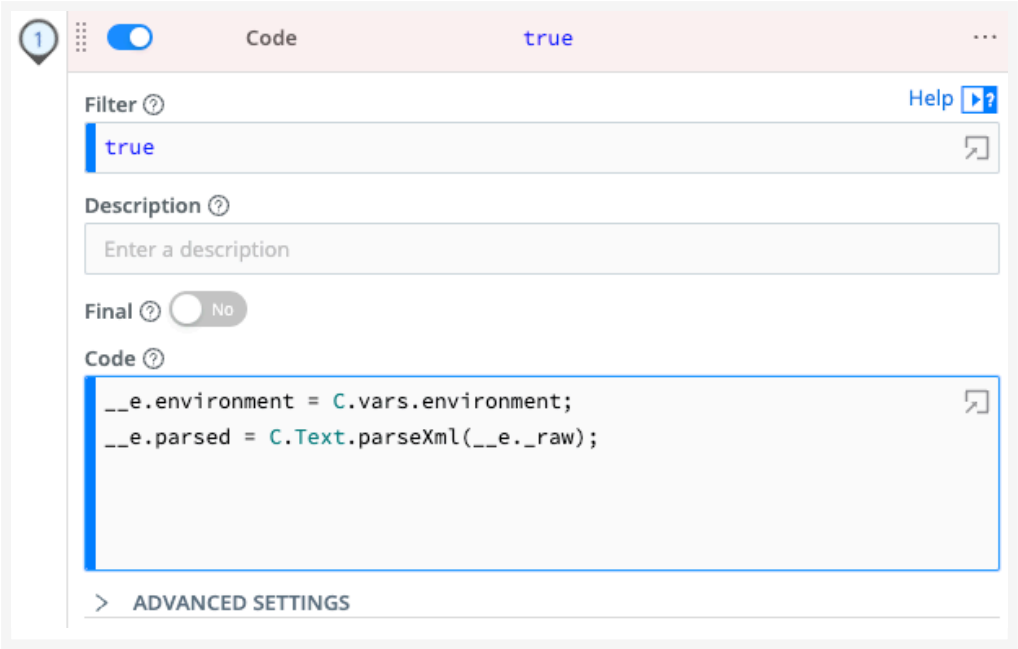
With some exceptions, the Code Function supports the options described in the following MDN JavaScript Guide topics:

- Expressions and Operators
- Global variables
- Control flow and error handling
- Loops and iteration
- Functions
- Numbers and dates
- Text formatting
- Regular Expressions
- Indexed collections
- Keyed collections

- Working with Objects

Access to `C.*` Global Object

From the Code's Function body, you can access global `C.*` object in Cribl Edge and its associated Cribl Expressions.



The global `C.*` object of a Code Function

Custom Functions

You can define your own functions to better organize your code. Both traditional and arrow functions are allowed.

Configure the Code Function

To configure this Function:

1. Add a Source and Destination to begin generating data. See Integrations for more information.
2. Add a new Pipeline or open an existing Pipeline. Add or capture sample data test your Pipeline as you build it. See Pipelines for more information.
3. At the top of the Pipeline, select **Add Function** and search for `Code`, then select it.
4. In the **Code** modal, configure the following general settings:
    - **Filter:** A JavaScript filter expression that selects which data to process through the Function. Defaults to `true`, meaning it evaluates all data events. If you don't want to use the default, you can

use a JavaScript expression to apply this Function to specific data events. See Build Custom Logic to Route and Process Your Data for examples.

- **Description:** A brief description of how this Function modifies your data to help other users understand its purpose later. Defaults to empty.
- **Final:** When enabled, stops data from continuing to downstream Functions for additional processing. Default: Off. Toggle **Final** on if the Code Function is the last Function in your Pipeline.
- **Code:** The JavaScript editor where you define your transformation logic. To interact with data in a Code Function, use the `__e` reserved variable. This variable represents the current event context. For example, to access a field named `field-name`, use the following bracket notation:

```
__e['field-name']
```

Copy

You can conceptualize your code as executing within a wrapper such as:

```
function(__e: Event) {
// your code here
}
```

Copy

See Usage Examples for specific code examples.

5. Configure additional settings as needed. For more information, see Advanced Settings.
6. Test that you configured the Function correctly by comparing sample incoming data with outgoing data in the Data Preview pane and Pipeline Diagnostics tool. See Troubleshoot the Code Function for additional information about troubleshooting your code.

Advanced Settings

Iteration limit: The maximum number of loops (iterations) allowed per execution of this Function. This safety boundary prevents infinite loops from hanging the system. Default: `5000`; Maximum: `10000`.

Error log sample rate: The frequency at which JavaScript runtime errors are written to the system log. For example, `1` logs every error, while `10` logs every tenth occurrence. Use a higher value to prevent log flooding during high-volume processing. Default: `1000`; Maximum: `1,000,000`.

Use unique log channel: When enabled, Cribl Edge creates a specific log channel for this instance: `func:code:${pipelineName}:${functionIndex}`. This is recommended for troubleshooting specific Pipelines. When toggled off, logs are sent to the generic `func:code` channel.

Usage Examples

The following examples use a common sample event representing CPU metrics. You can copy this JSON object into the Sample Data pane to test the code snippets and compare the outputs of different code operations.

Sample event input

```
{
  "cpus": [
    {"number": 1, "name": "CPU1", "value": 2.3},
    {"number": 2, "name": "CPU2", "value": 3.1},
    {"number": 3, "name": "CPU3", "value": 5.1},
    {"number": 4, "name": "CPU4", "value": 1.3}
  ],
  "arch": "Intel x64"
}
```

Copy

Filter Arrays

Use the `filter()` method to remove noise from an array by keeping only the elements that meet a specific condition. In this case, the goal is to only retain CPU objects where the value is greater than or equal to `3`. This allows for more fine-grained filtering by nested elements in an array.

```
// Create a new field containing only high-value CPU metrics
__e['cpus_filtered'] = __e['cpus'].filter(entry => entry.value >= 3);
```

Copy

The output now only contains events 2 and 3, which were greater than `3`:

```
{
  "cpus_filtered": [
    {"number": 2, "name": "CPU2", "value": 3.1},
    {"number": 3, "name": "CPU3", "value": 5.1}
  ]
}
```

Copy

Logic-Based Conditional Mapping

Use the Code Function to apply complex, range-based logic that would be difficult to manage in a standard Lookup table. This allows you to dynamically inject fields based on numeric thresholds or multiple combined conditions. To accomplish this, use a `map()` function with conditional logic to categorize data based on its value.

This code example adds severity or status levels to metrics dynamically:

```
// Add a status field to each CPU based on its load value
__e['cpus'] = __e['cpus'].map(entry => {
    let status = (entry.value > 4) ? 'CRITICAL' : 'NORMAL';
    return Object.assign(entry, { status });
});
```

Copy

The resulting output now has a status indicator based on its value:

```
{
  "cpus": [
    {"number": 1, "name": "CPU1", "value": 2.3, "status": "NORMAL"},
    {"number": 3, "name": "CPU3", "value": 5.1, "status": "CRITICAL"}
  ]
}
```

Copy

Modify Array Elements

Use the `map()` method to transform every element within an array. This example uses `Object.assign` to safely update the name field to lowercase for every CPU without losing other data points like `number` or `value`.

Code:

```
// Transform all CPU names to lowercase for naming consistency
__e['cpus'] = __e['cpus'].map(entry => Object.assign(entry, {'name': entry.name.toLow
```

Copy

The resulting output now uses lowercase name values instead of uppercase:

```
{
  "cpus": [
    {"number": 1, "name": "cpu1", "value": 2.3},
    {"number": 2, "name": "cpu2", "value": 3.1}
  ]
}
```

Copy

Troubleshoot the Code Function

Because the Code Function executes custom JavaScript, issues usually stem from script syntax, resource limits, or unintended logic loops. Use the following guide to identify and resolve common errors.

Identify Error Types

When a Code Function fails, Cribl Edge logs the error to the internal logs. You can view these by clicking the Logs tab in the Preview pane.

Debug

The Code Function provides a built-in `debug()` method to help you trace logic without flooding your production logs.

Messages logged by this `debug` helper function are shown in the Preview Log by default.

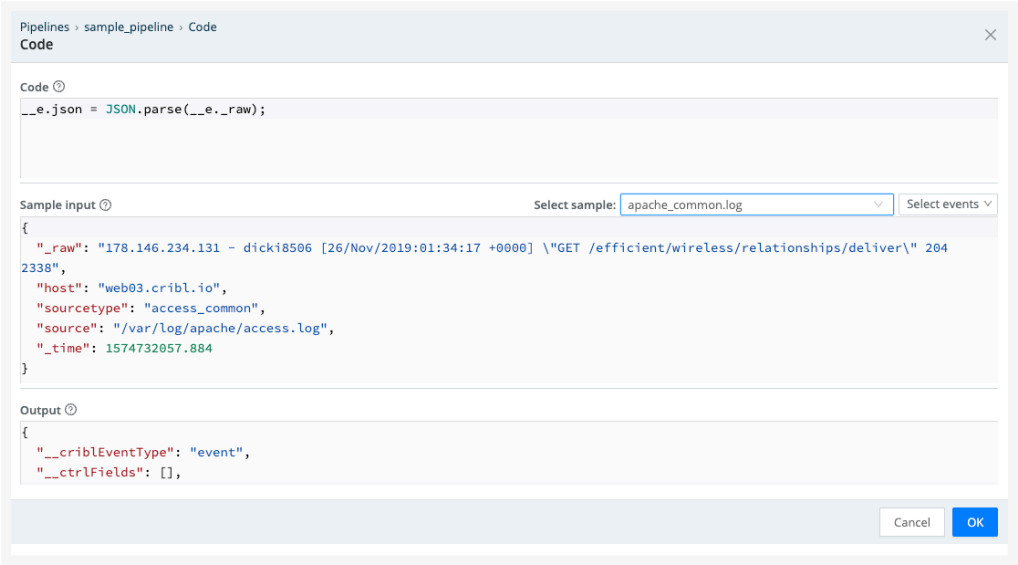
You can also route these messages to regular logs, although this requires setting the Function's log level (`func:code`) to `debug`.

```
// Example: Debugging a conditional branch
if (__e.status === 'CRITICAL') {
    debug('Processing a critical event', { eventId: __e._id });
    // your logic here
}
```

Copy

Preview

By using the expanded editor, you can run the expression against a sample event, and can preview the transformation:



Previewing a Code Function's transformation of an event

Partially Processed Events

If you notice that some events have new fields while others are missing expected transformations, you are likely encountering a runtime interruption. Because Cribl Edge does not roll back changes when a script fails, an error on line 5 will preserve any modifications made on lines 1 through 4.

To find where your script is failing, compare your code against an event that looks "half-processed." The point of failure is almost always the first missing field in your logic.

For example, take the following Code Function definition, which includes an `if` statement error in lines 6-8:

```
1    if (__e.name === 'Mario') {
2     __e.maybePlumber = true;
3    }
4    const MAX_LENGTH = 100;
5    // The if statement does not check for undefined/null values of address
6    if (__e.address.length > MAX_LENGTH) {
7     __e.addressTruncated = __e.address.slice(0, MAX_LENGTH);
8    }
9    __e.lastName = __e.lastName.toUpperCase();
```

Copy

When you run this example against the code:

```
const event1 = {
 name: 'Mario',
 lastName: 'Mario',
 address: '123 Plumber St, NYC'
};
```

Copy

It will produce the expected output:

```
const event1Processed = {
  name: 'Mario',
  lastName: 'MARIO',
  address: '123 Plumber St, NYC',
  maybePlumber: true
}
```

Copy

However, the code will fail when it receives this event, which has an undefined value for the `address` field:

```
const event2 = {
  name: 'Mario'
}
```

Copy

This event is only partially processed because the error occurred upstream of the address field logic. Since the Code Function executes line-by-line, the script terminated before reaching the instructions for the address field. The resulting output is incomplete:

```
const event2Processed = {
  name: 'Mario',
  lastName: 'Pablo',
  maybePlumber: true
};
```

Copy

Goat-to Resources

Cribl Concept - The Code Function

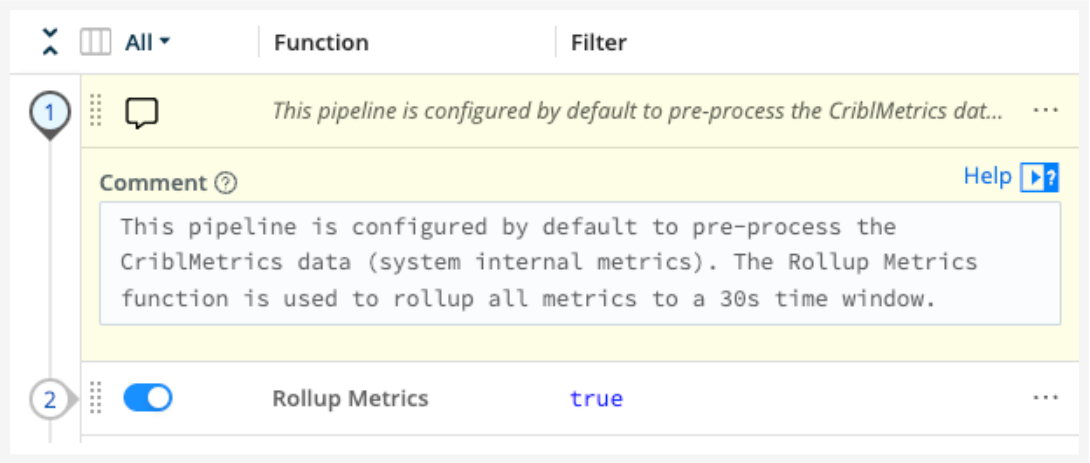
Video

15.8. Comment

The Comment Function adds a text comment in a Pipeline. It makes no changes to event data. The added comment is visible only within the Pipeline UI, where it is useful for labeling Pipeline steps. The Comment Function always has a minimum, non-zero processing time.

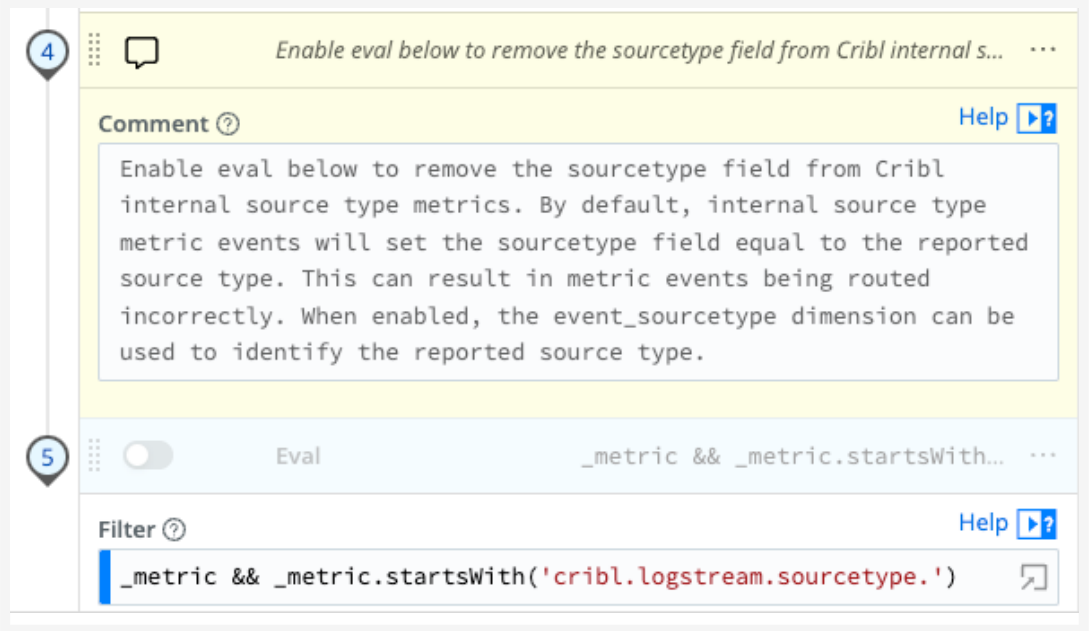
Usage

Comment: Add your comment as plain text in this field.



Examples

This comment labels the Pipeline's next function:



15.9. DNS Lookup

The DNS Lookup Function offers two operations useful in enriching security and other data:

- DNS lookups based on host name as text, resolving to A record (IP address) or to other record types.
- Reverse DNS Lookup. (This duplicates the behavior of Cribl Edge's prior Reverse DNS Function, which was deprecated as of v.2.4, and has been removed as of v.4.0.)

To reduce DNS lookups and minimize latency, the DNS Lookup Function incorporates a configurable DNS cache (including resolved and unresolved lookups). If you need additional caching, consider enabling OS-level DNS caching on each Cribl Edge Worker that will execute this Function. (OS-level caching options include DNSMasq, nscd, systemd-resolved, etc.)

Usage

Filter: Filter expression (JavaScript) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Simple description of this Function. Defaults to empty.

Final: Toggle on to stop feeding data to the downstream Functions. Default is toggled off.

DNS Lookup Fields Section

Lookup field name: Name of the field containing the domain to look up.

Resource record type: DNS record type (RR) to return. Defaults to `A`' record.

Output field name: Lookup result(s) will be added to this field. Leave blank to overwrite the original field specified in **Lookup field name**.

Reverse DNS Lookup Fields Section

Lookup field name: Name of the field containing the IP address to look up.



If the field value is not in IPv4 or IPv6 format, the lookup is skipped.

Output field name: Name of the field in which to add the resolved hostname. Leave blank to overwrite the original field specified in **Lookup field name**.

Fallback Resolving Methods

This section contains settings for resolving DNS short names.

Use /etc/resolv.conf: Toggle on if you want Cribl Edge to resolve DNS short names using the search or domain directive from `/etc/resolv.conf`. Default is toggled off. Not applicable for Windows hosts.

Use search or domain fallback(s): Add fallback value(s) for the DNS resolver to use when it cannot resolve a DNS short name.

Fall back to DNS.lookup(): Toggle on if you want Cribl Edge to make a `DNS.lookup()` call to resolve a DNS short name it can't resolve by other methods.

If you enable all of these methods at once. Cribl Edge will attempt to resolve short names by trying each method in the following order:

1. **Use /etc/resolv.conf**.
2. **Use search or domain fallback(s)**.
3. **Fall back to DNS.lookup()**.

If Cribl Edge can't resolve the short name after trying all of these methods, it will drop the output field from the event.



Potential Performance Impact

Making a `DNS.lookup()` call might degrade performance in unrelated areas of Cribl Edge. Use this method only if you have not been able to resolve DNS short names using other settings.

Advanced Settings

DNS server overrides: IP address(es), in RFC 5952 format, of the DNS server(s) to use for resolution. IPv4 examples: `1.1.1.1`, `4.2.2.2:53`. IPv6 examples: `[2001:4860:4860::8888]`, `[2001:4860:4860::8888]:1053`. If this field is not specified, Cribl Edge will use the system's DNS server.

Cache time to live (minutes): Determines the interval on which the DNS cache will expire, and its contents will be refetched. Defaults to `30` minutes. Use `0` to disable cache expiration/refresh behavior.

Cache size limit: Maximum number of DNS resolutions to cache locally. Before changing the default `5000`, consider the implications for your system. Higher values will increase memory usage. Highest allowed value is `100000`.

Log level for failed lookups: The log level at which to capture failed lookup events. Defaults to error.

Monitoring DNS Cache State

When the DNS Lookup Function succeeds in looking up a value it has stored in the DNS cache, that is called a **hit**. Each hit enables the Function to skip contacting a DNS server outside of Cribl Edge to perform the DNS lookup. The resulting performance improvement is the main benefit that the DNS cache provides.

When the DNS Lookup Function looks for a value in the DNS cache, but cannot find it, that is called a **miss**. In this case, the Function must contact a DNS server to perform the lookup.

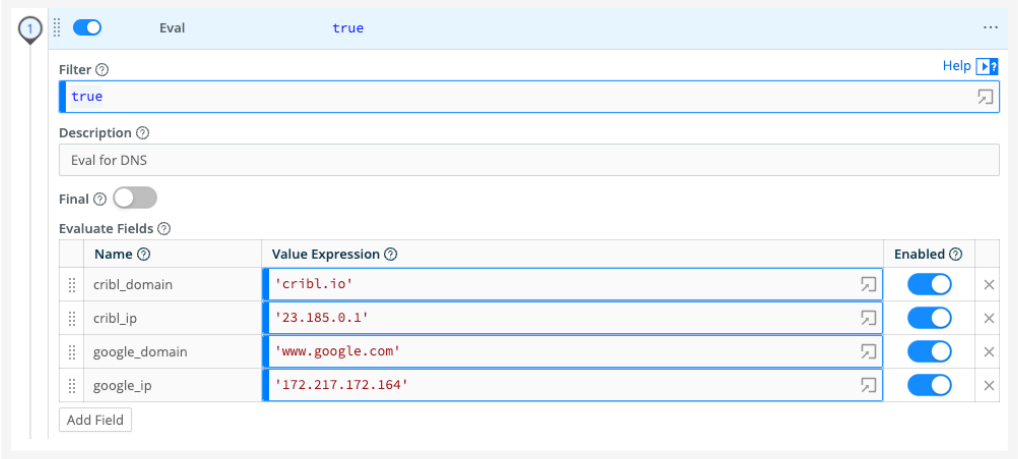
Cribl Edge logs the DNS cache's state in the `func:dns_lookup` channel. The table below explains some typical messages.

Log Message	Meaning
`hits`	Number of successful lookups from cache.
`misses`	Number of unsuccessful lookups from cache.
`keys`	Number of entries in cache.
`ksize`	Approximate amount of memory occupied by keys, in bytes.
`vsize`	Approximate amount of memory occupied by values, in bytes.

See Monitoring for a general explanation of logging in Cribl Edge.

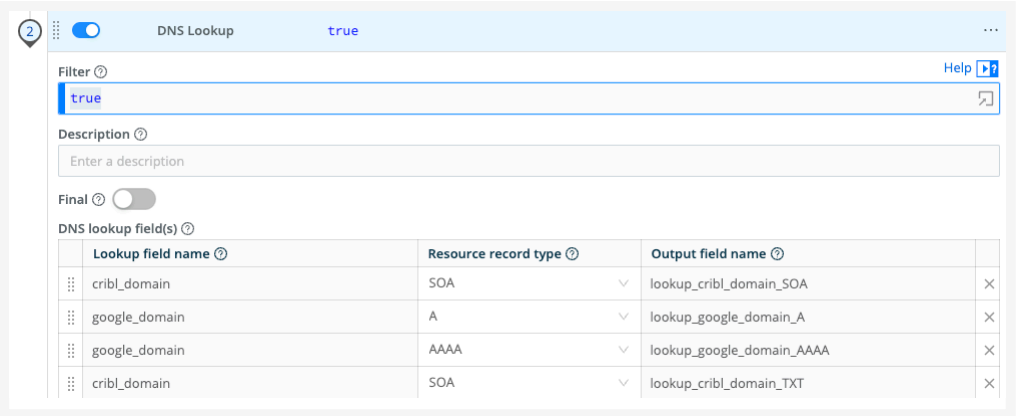
Example

This example Pipeline chains two Functions. First, we have an **Eval** Function that defines key-value pairs for two alphabetical domain names and two numeric IP addresses.



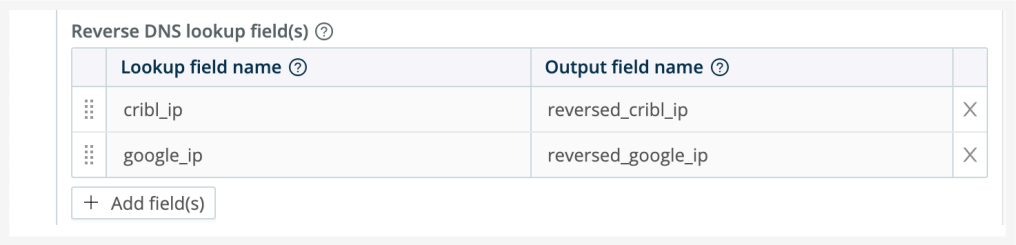
DNS Lookup: Eval Function

Next, the DNS Lookup Function looks up several record types for the two domain names, placing each retrieved record type in its own output field.



DNS Lookup: multiple record types

Finally, the same Function's **Reverse DNS lookup** section retrieves domain names for the two IP addresses.

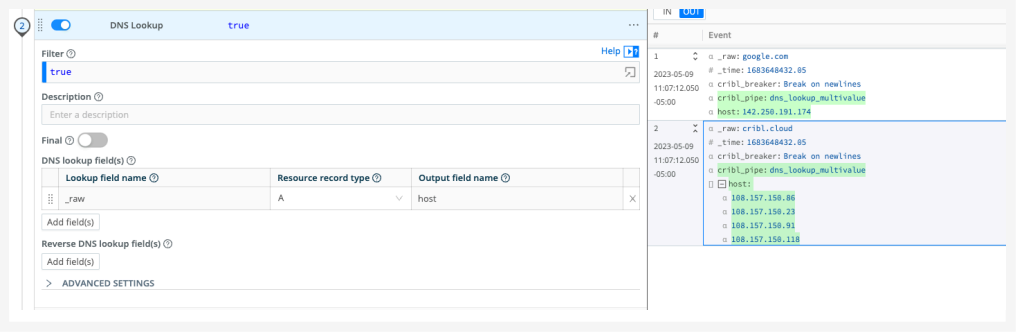


DNS Lookup: reverse lookups

Working with Multi-value Results

DNS records can contain single values mixed together with multi-value results. While this can be challenging to work with, the right code can adjust your results to the desired values.

In the screenshot below, the DNS lookup for `google.com` (event 1) returns a single record, while the `cribl.cloud` lookup (event 2) returns four addresses.



DNS lookup with multi-value result

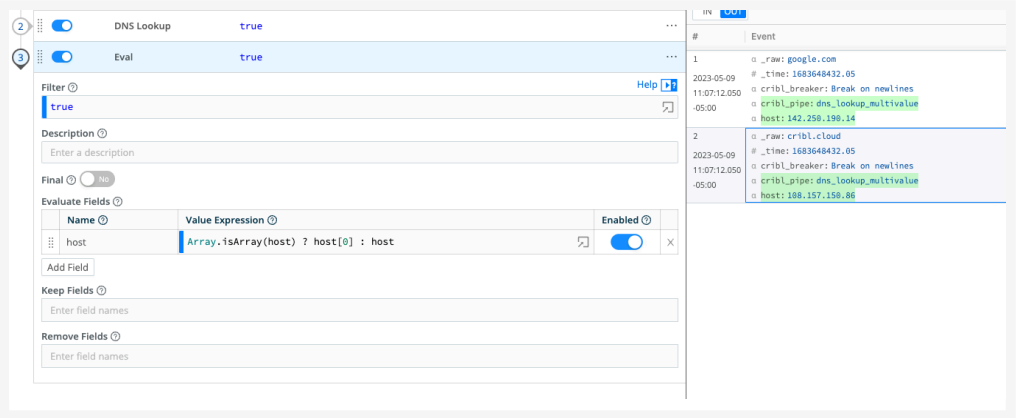
Sometimes a receiver is unable to consume the data in an array format. One solution to this problem is to pick the first result and convert it into a string. Here's a JavaScript eval to do this:

```
Array.isArray(host) ? host[0] : host;
```

Copy

This function checks whether the value of the `host` field is an array.

- If true, the result is the left side of the ternary expression - in this case, `host[0]`, which is the first element in the array, converted into a string. See event 2 in the screenshot below.
- If false, the result is the single-value string in the `host` field. See event 1.



DNS lookup multi-value result converted into a single value

Another option is to **randomly** pick an address from the result. To do this, use a modified version of the JavaScript eval:

```
Array.isArray(host) ? host[Math.floor(Math.random() * host.length)] : host
```

Copy

Here, instead of picking the first element, the code uses `Math.floor(Math.random() * host.length)` to randomly pick an element from the array.

15.10. Drop

The Drop Function deletes any events that meet its Filter expression. This is useful when you want to prevent certain events from continuing to a Pipeline's downstream Functions.

Usage

Filter: Filter expression (JavaScript) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Simple description about this Function. Defaults to empty.

Final: Toggle on to stop feeding data to the downstream Functions. Default is toggled off.

Examples

Scenario A:

Assume that we care only about errors, so we want to filter out any events that contain the word "success," regardless of case: "success," "SUCCESS," etc.

In our Drop Function, we'll use the JavaScript `search()` method to search the `_raw` field's contents for our target pattern. We know that `search()` returns a non-negative integer to indicate the starting position of the first match in the string, or `-1` if no match. So we can evaluate the Function as true when the return value is `>= 0`.

Filter: `_raw.search(/success/i)>=0`

Scenario B:

You can filter out specific JSON events based on their key-value pairs.

The following Filter expression uses a strict inequality operator to check, per event, whether `channel` has a different data type **or** value from `auth`. Matching events will drop, ensuring that only events with `"channel":"auth"` will pass to the next Function.

Filter: `channel !== 'auth'`

15.11. Drop Dimensions

The Drop Dimensions Function reduces the number of unique dimension combinations in outgoing metrics data to help address high cardinality. Use it to improve query performance and reduce storage costs by removing unnecessary dimensions.

This Function creates new aggregated events for the dropped dimensions. By dropping selected dimensions, you can reduce the number of unique time series, allowing you to group together incoming metrics and and roll them up into a single event. It merges frequently generated metrics into more manageable time windows, which creates smaller, coarser-grained datasets for downstream systems.

How the Drop Dimensions Function Works

The Drop Dimensions Function processes data in the following order:

1. **Filter:** Checks incoming data for events that match the specified filter values. You can filter by metric dimensions or any field on the incoming event.
2. **Drop:** Removes the specified dimensions from those events.
3. **Aggregate:** Combines all events with the same remaining fields down into a new aggregated event.
4. **Output:** Extracts, formats, and outputs the aggregated result as a new event.

Considerations

When using this Function, keep these considerations in mind:

- This Function does not perform any conversions from one metric type to another.
- Cribl Edge does not support aggregation of histogram, summary, or distribution metric types. The Function handles these metrics in special ways:
    - **Only unsupported metrics:** If an event only contains one of these unsupported metric types (histogram, distribution, or summary), these metrics pass through the Function unchanged.
    - **Mixed metrics:** If an event includes both supported and unsupported metric types (including histograms, distributions, or summaries), Cribl Edge splits the event into multiple events, each containing a single metric. These events are then processed by the Function normally.
- Cribl Edge creates new aggregated events for each supported metric. When you drop dimensions, it reduces the number of unique time series, enabling more events to be grouped and rolled up together. The result is a set of new, aggregated events that merge frequent metrics into more manageable time windows. Each new event includes only the fields specified in `__criblMetrics`:

- A field defined by `nameExpr`, where the value of the referenced field becomes the new field name. For example, if `nameExpr: ['_metric']` and the event has `_metric: 'cpu.usage'`, the output event will contain `cpu.usage: 1`.
- A value field as defined in the values array, where the referenced field provides the value of the new metric field. For example, if `values: ['_value']` and the event has `_value: 1`, then `1` will be the value for `cpu.usage`.
- Any matching dimension fields retained from the original event.



While `nameExpr` usually references a field (such as `_metric`), it can also be an expression. In such cases, Cribl Edge uses the evaluated result of the expression as the output field name.

Any fields that are on the event but not specified here will not be present on the outputted event.

Memory Usage and Cardinality

The Drop Dimensions Function processes metrics by grouping and aggregating data based on dimension combinations. This can lead to increased memory usage during processing if you are processing high-cardinality datasets where many unique combinations of metric names and dimensions exist.

To help identify and troubleshoot potential memory issues, the Function logs the following diagnostic values:

Diagnostic value	Description
`numUniqueMetricAndDims`	The number of unique combinations of metric names and dimensions currently being tracked.
`unFlushedEventCount`	The number of events still waiting to be aggregated based on their new (post-drop) dimension list.
`totalBytesTotal`	The total memory (in bytes) currently used by the Function to hold and process data.

If you observe unusually high values for `numUniqueMetricAndDims` or `totalBytesTotal`, this may be a sign of excessive cardinality. In extreme cases, high cardinality may cause the Drop Dimensions Function to exceed the memory limits of the Worker Processes. In these cases, consider dropping additional dimensions to reduce the number of unique time series generated.

Configure the Drop Dimensions Function

To configure this Function manually:

1. Add a Source and Destination to get data flowing. See Integrations for more information.

2. Add a new Pipeline or open an existing Pipeline. Add sample data or capture sample data to test your Pipeline as you build it. See Pipelines for more information.
3. At the top of the Pipeline, select **Add Function** and search for `Drop Dimensions`, then select it.
4. In the **Drop Dimensions** modal, configure the following general settings:
    - **Filter:** A JavaScript filter expression that selects which metrics to process through the Function. Defaults to `true`, meaning it evaluates all events. If you don't want to use the default, you can use a JavaScript expression to apply this Function to specific metric events. See Usage Examples: Filter Expressions for examples.
    - **Description:** A brief description of how this Function modifies your metrics to help other users understand its purpose later. Defaults to empty.
    - **Final:** When enabled, stops data from continuing to downstream Functions for additional processing. Default: Off. Toggle **Final** on if Drop Dimensions is the last Function in your Pipeline.
    - **Aggregation time window:** The duration of the tumbling window for aggregating events. Must be a valid time string (such as `10s`) and match the pattern `\d+[sm]$`.
    - **Dimensions to drop:** Add or edit the list of dimension names to drop. This field is not a JavaScript expression. Instead it uses a lightweight syntax that accepts simple strings and operators, including wildcard characters. For most dimension names, you can use the syntax `'dimensionName'` to access dimension names. See Usage Examples: Drop Dimensions for examples. Accepted formats:

Syntax	Description
`!string`	Exclude the specified dimension or field from the grouping. Example: `!proc`.
`string`	Include the specified dimension or field in the grouping. Example: `proc`.
`!*`	Exclude all dimensions or fields from the grouping.
`*`	Include all dimensions or fields in the grouping. **NOTE:** Use with caution. The wildcard character cannot search or filter for any dimensions or fields that use dot notation, such as `dimension.name` or `field.name`. StatsD and Open Telemetry use dot notation formats.
`string*` or `"string*"`	Include all dimensions or fields that begin with this string pattern.

    - **Flush on stream close:** When enabled (default), flushes aggregations when an input stream closes. When disabled, the **Aggregation time window** controls flush behavior. Disabling this setting may be preferable in cases such as:
        - Your input data consists of many small files.
        - You are sending data to Prometheus. Enabling Flush on stream close can cause Prometheus to receive multiple aggregations from the same Worker Process for the same time period.

Prometheus cannot distinguish between them and will ingest only the first one.

5. Test that you configured the Function correctly by comparing sample incoming data with outgoing data in the Data Preview pane and Pipeline Diagnostics tool. See Data Preview and Metrics View for more information.

Usage Examples

This section provides typical JavaScript expressions that you might commonly use with the Drop Dimensions Function.

Filter Expressions

The **Filter** setting is a JavaScript expression that selects which data to process through the Drop Dimensions Function. The following is an example of a JavaScript expression that filters for a metric named `proc.cpu_perc` that contains the dimensions `proc`, `pie`, and `unit` and other variants that may include these dimension names:

```
(_metric == 'proc.cpu_perc' || __criblMetrics[0].nameExpr.includes("'proc.cpu_perc'")
(_metric == 'proc.cpu_perc' || __criblMetrics[0].nameExpr.includes("'proc.cpu_perc'")
(_metric == 'proc.cpu_perc' || __criblMetrics[0].nameExpr.includes("'proc.cpu_perc'")
```

Copy

Drop Dimensions

You can use the **Drop dimension** setting to add or edit the list of dimension names to drop. This field is not a JavaScript expression. Instead it uses a lightweight syntax that accepts simple strings and operators, including wildcard characters.

For most dimension names, you can use the syntax `'dimensionName'` to access dimension names. For example:

```
'proc'
'pie'
'unit'
'!proc'
'proc*'
'unit*'
'!*'
'*'
```

Copy

15.12. Dynamic Sampling

The Dynamic Sampling Function filters out events based on an expression, a sample mode, and the volume of events. Your sample mode's configuration determines what percentage of incoming events will be passed along to the next step.



Each Worker Process executes this Function independently on its share of events. For details, see Functions and Shared-Nothing Architecture.

Usage

Filter: Filter expression (JavaScript) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Simple description about this Function. Defaults to empty.

Final: Toggle on to stop feeding data to the downstream Functions. Default is toggled off.

Sample mode: Defines how sample rate will be derived. For formulas and usage details, see Sample Modes below. Supported methods:

- Logarithmic (the default): `log(previousPeriodCount)`.
- Square root: `sqrt(previousPeriodCount)`.

Sample group key: Expression used to derive sample group key. For example: `${domain}:${httpCode}`. Each sample group will have its own derived sampling rate, based on the volume of events. Defaults to `` `${host}` ``.

All events without a host field passing through the Function will be associated with the same group and sampled the same.

Advanced Settings

- **Sample period Sec**: How often (in seconds) sample rates will be adjusted. Defaults to `30`.
- **Minimum events**: Minimum number of events that must be received, in previous sample period, for sampling mode to be applied to current period. If the number of events received for a sample group is less than this minimum, a sample rate of 1:1 is used. Defaults to `30`.

- **Sampling rate limit**. Maximum sampling rate. If the computed sampling rate is above this value, the rate will be limited to this value.

How Does Dynamic Sampling Work

Compared to static sampling, where users must first select a sample rate, Dynamic Sampling allows for **automatically adjusting** sampling rates, based on the volume of incoming events per sample group. This Function allows users to set only the aggressiveness/coarseness of this adjustment. Square Root is more aggressive than Logarithmic mode.

As an event passes through the Function, it's evaluated against the Sample Group Key expression to determine the sample group it will be associated with. For example, given an event with these fields: `...ip=1.2.3.42, port=1234...`, and a Sample Group Key of `` `${ip}:${port}` ``, the event will be associated with the `1.2.3.42:1234` sample group.



If the Sample Group Key is left at its `` `${host}` `` default, all events without a host will be associated with the same group and sampled the same.

When a sample group is new, it will initially have a sample rate of `1:1` for `Sample Period` seconds (this value defaults to 30 seconds). Once `Sample Period` seconds have elapsed, a sample rate will be derived based on the configured `Sample Mode`, using the sample group's event volume during the **previous** sample period.

For example, assuming a Logarithmic Sample Mode:

Period 0 (first 30s) – Number of events in sample group: `1000`, Sample Rate: `1:1`, Events allowed: `ALL` Sample Rate calculation for **next** period: `Math.ceil(Math.log(1000)) = 7`

Period 1 (next 30s) – Number of events in sample group: `4000`, Sample Rate: `7:1`: Events allowed: `572` Sample Rate calculation for **next** period: `Math.ceil(Math.log(4000)) = 9`

Period 2 (next 30s) – Number of events in sample group: `12000`, Sample Rate: `9:1`: Events allowed: `1334` Sample Rate calculation for **next** period: `Math.ceil(Math.log(12000)) = 10`

Period 3 (next 30s) – Number of events in sample group: `2000`, Sample Rate: `10:1`: Events allowed: `200` Sample Rate calculation for **next** period: `Math.ceil(Math.log(2000)) = 8`
...

Sample Modes

1. Logarithmic - The sample rate is derived, for each sample group, using a natural log: `Math.ceil(Math.log(lastPeriodVolume))`. This mode is **less aggressive**, and drops fewer events.

2. Square Root - The sample rate is derived, for each sample group, using: `Math.ceil(Math.sqrt(lastPeriodVolume))`. This mode is **more aggressive**, and drops more events.

Example

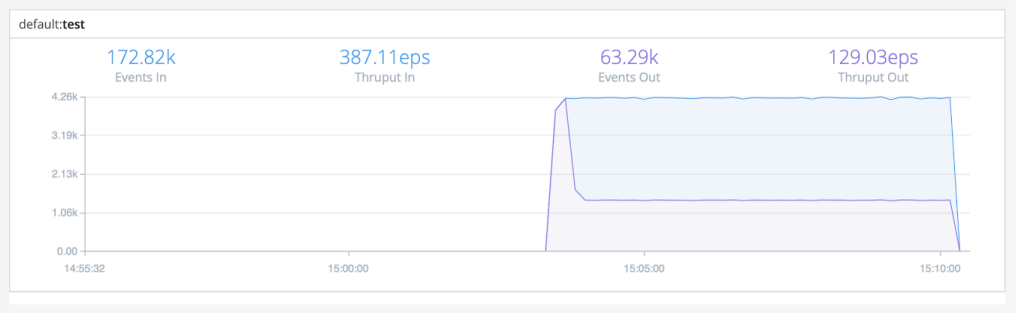
Here's an example that illustrates the effectiveness of using the Square Root sample mode.

Settings:

Sample Mode: `Square Root` Sample Period (sec): `20` Minimum Events: `3` Max. Sampling Rate: `3`

Results:

Events In: 4.23K Events Out: 1.41K



In this generic example, we reduced the incoming event volume from 4.23K to 1.41K. Your own results will vary depending on multiple parameters - the **Sample Group Key**, **Sample Period**, **Minimum Events**, **Max Sampling Rate**, and rate of incoming events.

For further examples, see Getting Smart and Practical With Dynamic Sampling.

See Also

- Sampling Function: Filter out events based on an expression and a sampling rate.

15.13. Eval

The Eval Function adds, modifies, or removes event fields. Eval is optimized for simple, single-value assignments and transformations using JavaScript expressions. For complex logic or multi-step processing, use the Code Function.

Usage

Filter: Filter expression (JavaScript) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Simple description about this Function. Defaults to empty.

Final: Toggle on to stop feeding data to the downstream Functions. Default is toggled off.

Evaluate fields: Table of event fields to evaluate and add/set. Select **Add Field** to add each field as a key-value pair, in a row with the following controls:

- **Name**: Enter the new (or modified) field's name.
- **Value Expression**: Enter a JavaScript expression to set the field's value - this can be a constant. Expressions can contain nested addressing. When you insert JavaScript template literals, strings intended to be used as values must be delimited with single quotes, double quotes, or backticks. For specific constraints, see Value Expression Limitations. For details on fields with non-alphanumeric characters, see Build Custom Logic to Route and Process Your Data: Fields with Non-Alphanumeric Characters.
- **Enabled**: Toggle this off to disable evaluating individual expressions, while retaining their configuration in the table. Useful for iterative development and debugging.

Keep fields: List of fields to keep in the event after processing by this Function. Takes precedence over the **Remove fields** setting. The field accepts wildcards (`*`), which supports both:

- Wildcards (`*`) to include multiple fields.
- Nested addressing, where you can reference a parent field and all its children using a wildcard. For example, if you convert `_raw` into an object, use `_raw*` to refer to `_raw` and all nested fields inside it. See Wildcard Lists for more information.

Remove fields: List of fields to remove from the event. Supports wildcards (*) and nested addressing. Cannot remove fields that match against the **Keep fields** list. Cribl Edge internal fields that start with `__` (double underscore) **cannot** be removed via wildcard. Instead, you must specify them individually. For example, `__myField` cannot be removed by specifying `__myF*`.

Value Expression Limitations

The **Value Expression** supports simple, single-value assignments and transformations. For advanced logic and complex operations, use the Code Function.

The following limitations apply to expressions:

1. **Reserved Words**: Expressions cannot utilize JavaScript **reserved words** like `constructor`, `prototype`, and `__proto__`.
   Cribl responds with: `Error: Expression cannot make use of the reserved word '<word>'.`
2. **Setters**: The definition or use of JavaScript **setters** is not allowed within expressions.
   Cribl responds with: `Error: Expression cannot use setter.`
3. **Assignment Operators**: Assignment operators like `=`, `+=`, `-=` are not permitted, except for direct assignment to the configured field. For example, `myField = <expression>`.
   Cribl responds with: `Error: Unallowed assignment operator found in expression.`
4. **Nested Scopes**: Expressions cannot introduce **nested scopes**. This includes, but is not limited to:
   - Loops (`for`, `while`, `do-while`)
   - Array iteration methods (`.map`, `.forEach`, `.reduce`, and so on)
   - Control structures (`if`, `switch`, `try-catch`)

   Cribl responds with: `Error: Expression cannot have nested scopes.`
5. **Internal Expressions**: Expressions are prevented from referencing internal Cribl expressions, such as `C.internal`.
   Cribl responds with: `Error: Expression cannot reference 'C.internal'.`
6. **Declarations**: Variable or function declarations are not allowed. This includes:
   - Variable declarations (`let`, `const`, `var`)
   - Function definitions (function declarations, arrow functions, function expressions)

   Cribl responds with: `Error: Expression cannot have any declarations.`

Using Keep and Remove

A field matching an entry in *both* **Keep** (wildcard or not) and **Remove** will *not* be removed. This is useful for implementing "remove all but" functionality. For example, to keep only `_time, _raw, source, sourcetype, host`, we can specify them all in **Keep**, while specifying `*` in **Remove**.

Negated terms are supported in both **Keep fields** and **Remove fields**. The list is order-sensitive when negated terms are used. Examples:

- `!foobar, foo*` means "All fields that start with 'foo' except `foobar`."

- `!foo*, *` means “All fields except for those that start with ‘foo’.”

Examples

Note that Functions use the special variable `__e` to access the `(context)` event inside JavaScript expressions.

Scenario A: Create field `myField` with static value of `value1`:

- **Name:** `myField`
- **Value Expression:** `'value1'`

Scenario B: Set field `action` to `blocked` if `login==error`:

- **Name:** `action`
- **Value Expression:** `login=='error' ? 'blocked' : action`

Scenario C: Create a multivalued field called `myTags` (array):

- **Name:** `myTags`
- **Value Expression:** `['failed', 'blocked']`

Scenario D: Add value `error` to the multivalued field `myTags`:

- **Name:** `myTags`
- **Value Expression:** `login=='error' ? [...myTags, 'error'] : myTags`

(The above expression is literal, and uses JavaScript spread syntax.)

Scenario E: Rename an `identification` field to the shorter `ID` - copying over the original field’s value, and removing the old field:

- **Name**: `ID`
- **Value Expression**: `identification`
- **Remove Field**: `identification`

See Also

For more usage examples, see these Better Practices topics:

- Ingest-time Fields, which demonstrates both adding and removing fields.
- Reducing Windows XML Events, which demonstrates using Eval with the `C.Text.parseWinEvent` expression method to parse `_raw` Windows XML events into prettified JSON objects.

Usage Notes

Consider the following when working with Eval Functions.

Create Parent Objects First

Before you can use the Eval function on a new child object, you must create the parent object - then define the children using Eval Functions.

Example:

```
parent  =  (parent || { child1: child1Value })
parent.child2 = child2Value
parent.child3 = child3Value
<some other Evals, if you need them>
```

Copy

To Append:

```
parent =  Object.assign(parent, { child2: child2Value })
```

Copy

To Create a New Field:

```
parent2 =  Object.assign(parent, { child2: child2Value, child3: child3Value })
```

Copy

Execution Without Assignment

The Eval Function can execute expressions **without** assigning their value to the field of an event. You can do this by simply leaving the left-hand side input empty, and having the right-hand side do the assignment.

Example: Parse and Merge to Existing Field

`Object.assign(foo, JSON.parse(bar), JSON.parse(baz))` on the right-hand side (and left-hand side empty) will JSON-parse the strings in `bar` and `baz`, merge them, and assign their value to `foo`, an already existing field.

Example: Reference Event with `__e`

To parse JSON, enter `Object.assign(__e, JSON.parse(_raw))` on the right-hand side (and left-hand side empty). `__e` is a special variable that refers to the `(context)` event **within** a JavaScript expression. In this case, content parsed from `_raw` is added at the top level of the event.

15.14. Event Breaker Function

This Function enables you to split large blobs or streams of events into discrete events within a Pipeline. This is useful for Sources like Azure Event Hubs, which do not natively support Event Breakers.

Even with Sources that do support Event Breakers (like Raw HTTP), it sometimes makes sense to use both a Source-configured Event Breaker on the incoming stream, and an Event Breaker Function within a Pipeline.

Limitations



The Event Breaker Function operates only on data in `_raw`. For other events, move the array to `_raw` and stringify it before applying this Function.

The largest event that this Function can break is about about 128 MB (`134217728` bytes). Events exceeding this maximum size will be split into separate events, but left unbroken. Cribl Edge will set these events' `__isBroken` internal field to `false`.

Unlike regular Event Breakers, Event Breaker Functions do not have names, only descriptions.

Usage

Use the following options to define this Event Breaker Function.

Filter: Filter expression (JavaScript) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Simple description of this Function. Defaults to empty.

Final: Toggle on to stop feeding data to the downstream Functions. Default is toggled off.

Existing or new?: Whether to use an existing ruleset or create a new one. Defaults to `Use Existing`.

- When **Existing or new?** is set to `Use Existing`, the **Existing ruleset** drop-down appears and you choose a ruleset from there.
- When **Existing or new?** is set to `Create New`, the ruleset creation UI appears and you configure its settings.

Advanced Settings

Add to cribl_breaker: Whether to add the `cribl_breaker` field to output events. Defaults to toggled on.

How this field behaves depends on whether you are **also** using a regular Event Breaker with your Source. That matters because a regular Event Breaker **always** adds the `cribl_breaker` field to events, which it does **before** the data reaches your Event Breaker Function.

When you are **not** using a regular Event Breaker, there is no `cribl_breaker` field yet when the data reaches your Event Breaker Function. At this point, `cribl_breaker` is added, and:

- When **Existing or new?** is set to `Use Existing`, `cribl_breaker`'s value is set to the name of the existing ruleset you chose.
- When **Existing or new?** is set to `Create New`, the `cribl_breaker`'s value is set to `"event_breaker_func"`. Since Event Breaker Functions do not have names, the string `event_breaker_func` serves as a kind of generic name that represents whatever new Event Breaker Function you created.

When you **are** also using a regular Event Breaker, the `cribl_breaker` field already exists when the data reaches your Event Breaker Function. At this point, the value of `cribl_breaker` is changed from a string to an array, with the first item in the array being the value originally set by the regular Event Breaker, and the second item determined as follows:

- When **Existing or new?** is set to `Use Existing`, the second item's value is set to the name of the existing ruleset you chose.
- When **Existing or new?** is set to `Create New`, the second item's value is set to `"event_breaker_func"`. Since Event Breaker Functions do not have names, the string `event_breaker_func` serves as a kind of generic name that represents whatever new Event Breaker Function you create.



Cribl Edge truncates timestamps to three-digit (milliseconds) resolution, omitting trailing zeros.

In Cribl Edge 3.4.2 and above, where an Event Breaker Function has set an event's `_time` to the current time – rather than extracting the value from the event itself – it will mark this by adding the internal field `__timestampExtracted: false` to the event.

Examples

Handling syslog data and nested JSON data are two primary use cases for Event Breaker Functions.

Event Breaker Functions for syslog Data

Event Breaker Functions can help you deal with syslog data that does not break correctly. Examples include multi-line syslog data, and, the kinds of non-standard syslog data emitted by Blue Coat proxy appliances,

Layer7 API Gateways, and other appliances. See the Cribl video Scaling syslog for an in-depth discussion.

(In this context, “non-standard” means syslog data that only partly conforms to the standard Syslog Protocol defined in RFC 5424, or its predecessor, the BSD syslog Protocol defined in RFC 3164.)

Event Breaker Functions for Nested JSON Data

Kafka-based data from Confluent, Azure Event Hubs, Google Pub Sub, or Kafka itself, is nicely structured as events according to the Kafka protocol. But when these events contain JSON objects which you want to break into their constituent objects, you can use an Event Breaker Function to do that.

Troubleshooting

If you notice fragmented events, check whether Cribl Edge has added a `__timeoutFlush` internal field to them. This diagnostic field’s presence indicates that the events were flushed because the Event Breaker buffer timed out while processing them. These timeouts can be due to large incoming events, backpressure, or other causes.

15.15. Flatten

The Flatten Function pulls fields out of a nested structure, moving nested key-value pairs (fields) up to a higher level in the object. You can specify:

- Individual fields to flatten.
- The depth at which to flatten fields.
- The delimiter to use when concatenating keys.
- An optional prefix to add to transformed field names.



The Flatten Function creates fully qualified names for promoted fields. If you simply need to promote fields without transforming their names, use the `eval` Function.

Usage

Filter: Filter expression (JavaScript) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Simple description of this Function. Defaults to empty.

Final: Toggle on to stop feeding data to the downstream Functions. Default is toggled off.

Fields: List of top-level fields to include for flattening. Defaults to an empty array, which means all fields. Limit to specific fields by typing in their names, separated by hard returns. Supports wildcards (`*`). Supports double-underscore (`__`) internal fields only if individually enumerated - not via wildcards.

Prefix: Prefix string for flattened field names. Defaults to empty.

Depth: Number representing the nested levels to consider for flattening. Minimum `1`. Defaults to `5`.

Delimiter: Delimiter to use for flattening. Defaults to `_` (underscore).

Example

Add the following test sample in **Preview** > **Paste a Sample**:

```
input
```

```
{
        "accounting": [{
                "firstName": "John",
                "lastName": "Doe",
                "age": 23
        }, {
                "firstName": "Mary",
                "lastName": "Smith",
                "age": 32
        }],
        "sales": [{
                "firstName": "Sally",
                "lastName": "Green",
                "age": 27
        }, {
                "firstName": "Jim",
                "lastName": "Galley",
                "age": 41
        }]
}
```

Copy

Under **Select Event Breaker**, choose **ndjson** (newline-delimited JSON), and select **Save as a Sample File**.

Here's sample output with all settings at default:

output

```
{
  "accounting_0_firstName": "John",
  "accounting_0_lastName": "Doe",
  "accounting_0_age": 23,
  "accounting_1_firstName": "Mary",
  "accounting_1_lastName": "Smith",
  "accounting_1_age": 32,
  "sales_0_firstName": "Sally",
  "sales_0_lastName": "Green",
  "sales_0_age": 27,
  "sales_1_firstName": "Jim",
  "sales_1_lastName": "Galley",
  "sales_1_age": 41,
```

15.16. Fold Keys

The Fold Keys Function transforms flat field names that include a common separator (such as `.` or `/`) into a more logical, nested structure. Use this Function to normalize hierarchical data that expresses logical relationships through field names. The Fold Keys Function restructures the field names into grouped, nested objects without changing their values.

Use this Function to:

- **Normalize data from systems that encode hierarchy in field names:** Standardize keys from sources that encode similar data differently. For example, you could transform keys such as `http.status`, `http.method`, and `http.url` into a nested structure under `http`.
- **Reduce field name redundancy for more efficient storage and easier parsing:** Group similar fields under a parent key to reduce clutter and improve readability. With nested structures also give you the option to apply logic to an entire group (`http.*`) rather than individual fields using another Function in your Pipeline.
- **Improve schema compatibility or prepare data for JSON-based Destinations or downstream receivers that expect nested objects:** Structure your data for downstream consumption by tools such as Elasticsearch or certain APM platforms that expect hierarchical data structures for querying or visualizing.

See Usage Examples for an in-depth example of how this Function processes data.

How the Fold Keys Function Works

The Fold Keys Function is a simpler version of the Eval Function, designed specifically to group related fields by a common separator in their names. If you're familiar with Cribl Search, the Function behaves like the foldkeys operator, but for use within Cribl Edge.

The Fold Keys Function processes data in the following order:

1. **Search:** It looks for field names that contain the specified separator string (such as a period or underscore). If a field name does not include the separator, the Function skips it and passes it through unchanged.
2. **Group:** For each matching field name, the Function splits the name into parts based on the separator. It uses these parts to organize related fields under a shared top-level key.
3. **Nest:** The Function creates a nested data structure schema based on the grouped parts. For example, a field like `http.status.code` would become:

```
"http": {
  "status": {
    "code": 404
    }
  }
```

Copy

See Usage Examples for a practical, real-world example of how this Function processes data.

Configure the Fold Keys Function

To configure this Function:

1. Add a Source and Destination to begin generating data. See Integrations for more information.
2. Add a new Pipeline or open an existing Pipeline. Add or capture sample data test your Pipeline as you build it. See Pipelines for more information.
3. At the top of the Pipeline, select **Add Function** and search for `Fold Keys`, then select it.
4. In the **Fold Keys** modal, configure the following general settings:
    - **Filter:** A JavaScript filter expression that selects which data to process through the Function. Defaults to `true`, meaning it evaluates all data events. If you don't want to use the default, you can use a JavaScript expression to apply this Function to specific data events. See Build Custom Logic to Route and Process Your Data for examples.
    - **Description:** A brief description of how this Function modifies your data to help other users understand its purpose later. Defaults to empty.
    - **Final:** When enabled, stops data from continuing to downstream Functions for additional processing. Default: Off. Toggle **Final** on if Fold Keys is the last Function in your Pipeline.
    - **Delete original:** When toggled on (default), the data output only includes the folded field names, meaning it removes the original data from the raw location. When toggled off, the data output includes both the folded and original field names.
    - **Separator string:** Set a separator character to use as the path separator. For example: `/` or `_`. This separator determines how to group the fields. Defaults to `.` (dot).
    - **Selection regular expression (optional):** Input a regular expression. If defined, the Function only groups field names that match this expression, along with the separator string.
5. Test that you configured the Function correctly by comparing sample incoming data with outgoing data in the Data Preview pane and Pipeline Diagnostics tool. See Data Preview for more information.

Usage Examples

The following is an example of raw data before applying the Fold Keys Function:

```
{
  "_raw": "Mar 27 23:45:05 PA-VM 2020/05/07 02:40:14,44A1B3FC68F5304,TRAFFIC,end,192.
  "_time": 1743144305,
  "source": "cribl",
  "data_metrics_bytes": 12345678,
  "data_metrics_count": 987,
  "data_metrics_latency_p95": 123,
  "data_metrics_latency_p90": 56,
  "host_ip": "203.0.113.0"
}
```

Copy

In this example, there are multiple fields that begin with `data_metrics`. You could use the Fold Keys Function to group these field names together into a more logical, nested data structure.

In this example, the Fold Keys Function uses these settings:

- **Delete original:** Toggled on
- **Separator string:** `_`
- **Selection regular expression:** `^data`

Together, these settings nest all fields that start with `data_` and creates two hierarchies using the underscore `_` character as the separator string. The Regular Expression `^data` matches any string that starts with the word `data`.

After the Fold Key Function processes the data, the output looks like this:

```
{
  "_raw": "Mar 27 23:45:05 PA-VM 2020/05/07 02:40:14,44A1B3FC68F5304,TRAFFIC,end,192.
  "_time": 1743144305,
  "source": "cribl",
  "data": {
    "metrics": {
      "bytes": 12345678,
      "count": 987,
      "latency": {
        "p95": 123,
```

15.17. GeoIP

The GeoIP Function enriches events with geographic fields, given an IP address. It works with MaxMind's GeoIP or IPinfo's IP address binary database.

Prerequisite

You need to host the `.mmdb` database file. The following steps cover this process at a high level. They link to our Lookup Geographic Data from IP Addresses topic, where you can find additional details.

1. Download and extract the `.mmdb` database file.
2. Determine where to place the database file. We recommend the `$CRIBL_HOME/state/` subdirectory, which is already listed in the default `.gitignore` file that ships with Cribl Edge.
   You always have the option to upload the file to Cribl Edge's Lookups Library. In a Cribl.Cloud deployment, this is currently the only option. For details, see About Lookups.
3. Place the database file on your Edge Node(s). In a distributed deployment, we recommend having the file on the Leader and all Edge Nodes. Smaller deployments can get away with hosting the file only on the Leader Node.
4. Optionally, set up automatic updates to the MaxMind and IPinfo database files.

Usage

Filter: Filter expression (JavaScript) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Simple description about this Function. Defaults to empty.

Final: Toggle on to stop feeding data to the downstream Functions. Default is toggled off.

GeoIP file (.mmdb): Path to the database, in binary format, with `.mmdb` extension.



If the database file is located within the lookup directory (`$CRIBL_HOME/data/lookups/`), the **GeoIP file** does not need to be an absolute path.

In distributed deployments, ensure that the database file is in the same location on the Leader Node and all Edge Nodes. However, if you've uploaded `.mmdb` files via Cribl Edge's Lookups Library UI, just select this box to display and select them on a drop-down list. Your selection here will handle the path management automatically.

IP field: Field name in which to find an IP to look up. Can be nested. Defaults to `ip`.

Result field : Field name in which to store the GeoIP lookup results. Defaults to `geoip`.

Examples

Assume that you are receiving SMTP logs, and need to see geolocation information associated with IPs using the SMTP service.

Here's a sample of our data, from IPSwitch IMail Server logs:

```
03:19 03:22 SMTPD(00180250) [192.168.1.131] connect 74.136.132.88 port 2539 03:19 03:22 SMTPD(00180250) [74.136.132.88] EHLO msnbc.com 03:19 03:22 SMTPD(00180250) [74.136.132.88] MAIL FROM:<info-jjgcdshx@test.us> 03:19 03:22 SMTPD(00180250) [74.136.132.88] RCPT To:<user@domain.com>
```

In this example, we'll chain together three Functions. First, we'll use a Regex Extract Function to isolate the host's IP. Next, we'll use the GeoIP Function to look up the extracted IP against our geoIP database, placing the returned info into a new `__geoip` field. Finally we'll use an Eval Function to parse that field's city, state, country, ZIP, latitude, and longitude.

Function 1 - Regex Extract

Regex: `\[(?<ip>\S+)\]` Source field: `_raw` Result: `74.136.132.88`

Function 2 - GeoIP

Event's IP field: `ip` Result field: `__geoip`

Function 3 - Eval

Name	Value Expression
City	`__geoip.city.names.en`
Country	`__geoip.country.names.en`
Zip	`__geoip.postal.code`
Lat	`__geoip.location.latitude`
Long	`__geoip.location.longitude`

In the Eval Function's **Remove fields** setting, you could specify the `__geoip` field for removal, if desired. However, its `__` prefix makes it an internal field anyway.



For a hosted tutorial on applying the GeoIP Function, see Cribl's GeoIP and Threat Feed Enrichment Sandbox.

15.18. Grok

The Grok Function extracts structured fields from unstructured log data, using modular regex patterns.

Usage

Filter: Filter expression (JavaScript) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Optional description of this Function's purpose in this Pipeline. Defaults to empty.

Final: Toggle on to stop feeding data to the downstream Functions. Default is toggled off.

Pattern: Grok pattern to extract fields. You can use a Grok pattern from any of the configured files. Click the Expand button at right to open a preview/validation modal.
Syntax supported: `%{PATTERN_NAME:FIELD_NAME}`.

Select **Add pattern** to chain more patterns.

Source field: Field on which to perform Grok extractions. Defaults to `_raw`.

Management

You can add and edit Grok patterns via Cribl Edge's UI by selecting **Knowledge** > Grok Patterns. Pattern files are located at: `$CRIBL_HOME/(default|local)/cribl/grok-patterns/`

Example

Example event:

```
{"_raw": "2020-09-16T04:20:42.45+01:00 DEBUG This is a sample debug log message"}
```

Copy

Pattern: `%{TIMESTAMP_ISO8601:event_time} %{LOGLEVEL:log_level} %{GREEDYDATA:log_message}`
Source Field: `_raw`

Event after extraction:

```
{"_raw": "2020-09-16T04:20:42.45+01:00 DEBUG This is a sample debug log message",
  "_time": 1600226442.045,
  "event_time": "2020-09-16T04:20:42.45+01:00",
  "log_level": "DEBUG",
  "log_message": "This is a sample debug log message",
}
```

Copy

Note the new fields added to the event: `event_time`, `log_level`, and `log_message`.

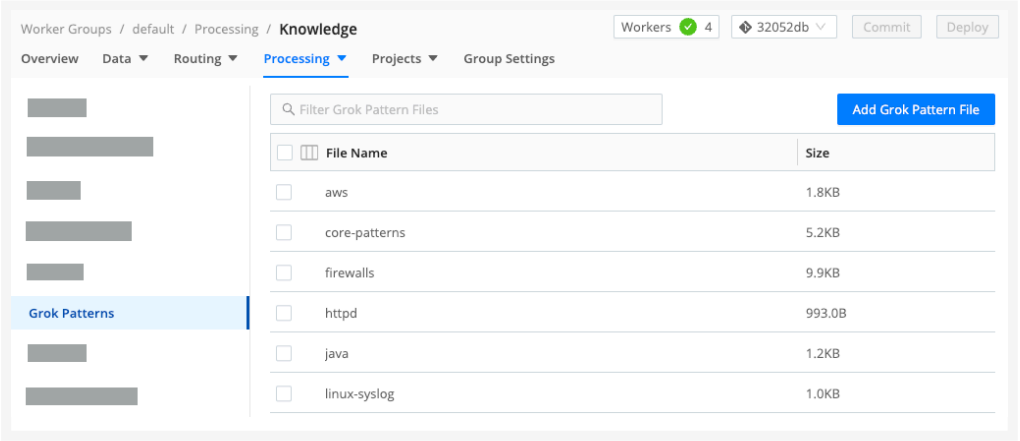
References

- Syntax for a Grok pattern is `%{PATTERN_NAME:FIELD_NAME}`. Example: `%{IP:client} %{WORD:method}`.
- Useful link for creating and testing Grok patterns: http://grokconstructor.appspot.com
- Additional patterns are available here: https://github.com/logstash-plugins/logstash-patterns-core/tree/main/patterns

Available Grok Patterns

Cribl Edge ships common Grok Patterns for basic scenarios that you can use with the Grok Function, organized as a collection of files.

You can refer to a single pattern in multiple Grok Functions in different Pipelines. When editing them, be aware that any change will affect all places it is used in.



Available Grok Patterns

To find the patterns, select **Fleets** from the sidebar, and choose a Fleet. Then, on the **Fleets** submenu, select **More**, then **Knowledge**, then **Grok Patterns**.

Add a Grok Pattern File

To add a new Grok pattern file, follow these steps:

1. In the sidebar, select **Fleets** from the sidebar, and choose a Worker Group or Fleet.
2. On the **Fleets** submenu, select **More**, then **Knowledge**, then **Grok Patterns**.
3. Select **Add Grok Pattern File**.
4. Enter a unique **File name**.
5. Paste the content of the Grok file into **Content**.
6. Select **Save**.

15.19. JSON Unroll

The JSON Unroll Function processes a JSON object string in the `_raw` field. It unrolls (explodes) an array of objects within this JSON into individual events, while inheriting the top-level fields from the original JSON object. This function is ideal for scenarios where processing individual items within an array is required while retaining the context provided by parent fields.

For efficiency, we recommend avoiding the JSON Unroll Function for certain types of data, such as CloudTrail, Office 365, or Kubernetes CloudWatch logs, which are often collected as JSON arrays. Instead, use an Event Breaker for inputs that support it. However, Event Breakers like JSON Array do not retain parent fields from the original JSON object. If retaining parent fields is critical, use the JSON Unroll Function or use a pre-processing Pipeline to extract parent fields before breaking events.

How the Function Works

When JSON Unroll runs, it parses the JSON object string in `_raw` into an internal field named `__json`. This internal field contains the full parsed structure of the event, including the array that you configure the Function to unroll. For example, if your array field is named `data`, the parsed JSON is available as `__json.data`, which you can reference in downstream Functions without reparsing `_raw`.

JSON Unroll then:

1. Locates the array at the configured **Path** within `__json`.
2. Iterates over each element in that array.
3. For each element, creates a new event that:
    - Inherits the top-level fields from the original JSON object. For example, `date`, `name`, or `age`.
    - Adds the current array element under the field specified in **New name**, or under its original field name if **New name** is left empty.
    - Serializes the resulting object back into `_raw` for the new event.

To inspect the parsed JSON and verify how JSON Unroll is working, enable **Show Internal Fields** in Data Preview. This exposes `__json` and other internal fields so you can see the parsed structure the Function is operating on.

Configuration

Filter: Filter expression (JavaScript) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Simple description about this Function. Defaults to empty.

Final: Toggle on to stop feeding data to the downstream Functions. Default is toggled off.

Path: Path to array to unroll, such as `foo.0.bar`. This path is resolved against the parsed JSON stored in `__json`.

New name: The name that the exploded array element will receive in each new event. Leave empty to expand the array element with its original name.

Example

Assume you have an incoming event that has a `_raw` field as a JSON object string like this:

Sample _raw field

```
{"date":"9/25/18 9:10:13.000 PM",
   "name":"Amrit",
   "age":42,
   "allCars": [
       { "name":"Ford", "models":[ "Fiesta", "Focus", "Mustang" ] },
       { "name":"GM", "models":[ "Trans AM", "Oldsmobile", "Cadillac" ] },
       { "name":"Fiat", "models":[ "500", "Panda" ] },
       { "name":"Blackberry", "models":[ "KEY2", "Bold Touch 9900" ] }
   ]
}
```

Copy

With the following settings configured:

- **Path**: `allCars`
- **New Name**: `cars`

Your output events would look like:

Resulting Events

```
Event 1:
{"_raw":"{"date":"9/25/18 9:10:13.000 PM","name":"Amrit","age":42,"cars":{"name":"For

Event 2:
{"_raw":"{"date":"9/25/18 9:10:13.000 PM","name":"Amrit","age":42,"cars":{"name":"GM"

Event 3:
{"_raw":"{"date":"9/25/18 9:10:13.000 PM","name":"Amrit","age":42,"cars":{"name":"Fia

Event 4:
{"_raw":"{"date":"9/25/18 9:10:13.000 PM","name":"Amrit","age":42,"cars":{"name":"Bla
```

Copy

Each element under the original **allCars** array is now placed in a **cars** field in its own event, inheriting original top level fields: **date**, **name**, and **age**. The parsed representation of each output event is also available under the internal `__json` field, which you can use in downstream Functions instead of reparsing `_raw`.

See Also

- The Cribl Knowledge Pack provides sample Pipelines that demonstrate converting a JSON string into an object literal, and validating JSON data against a schema.

Troubleshoot High Memory Usage or Slow Throughput

Issue: Pipelines are sluggish or Edge Nodes are hitting memory limits when processing Azure flow data.

Cause: Using the JSON Unroll Function on highly aggregated logs (like Azure VNet Flow logs) forces the Pipeline to hold massive objects in memory before they are split.

Solution: Migrate the unrolling logic from the Pipeline to the Source by applying the Azure VNet Flow Event

15.20. Lookup

The Lookup Function enriches events with external fields, using lookup table files in CSV, compressed `.csv.gz`, or binary `.mmdb` format.



CSV files are text-based and untyped. This means all data within the file is treated as strings, regardless of its intended type (like, numbers or dates).

Usage

Filter: Filter expression (JavaScript) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Simple description of this Function. Defaults to empty.

Final: Toggle on to stop feeding data to the downstream Functions. Default is toggled off.

Lookup file path (.csv, .csv.gz): Path to the lookup file. Select an existing file that you've uploaded via Cribl Edge's UI at **Knowledge > Lookups**, or specify the path. You can reference environment variables via `$`, such as `$CRIBL_HOME/file.csv`.



When you configure this field via a distributed deployment's Leader Node, Cribl Edge will swap `$CRIBL_HOME/groups/<groupname>/` for `$CRIBL_HOME` when validating whether the file exists. In this case, the default upload path changes from `$CRIBL_HOME/data/lookups` (single-instance deployments) to `$CRIBL_HOME/groups/<groupname>/data/lookups/` (distributed deployments).

Use Disk-Based Lookup: Default: off. Toggle on if the lookup file referenced in the **Lookup file path** is a disk-based lookup. When toggled on, this setting removes the **Match Mode** and **Reload Period (sec)** options, as they do not apply to disk-based lookups.



Cribl Edge does not support disk-based lookups, only in-memory lookups.

Match mode: Defines the format of the lookup file, and indicates the matching logic that will be performed. Defaults to `Exact`. This option is only available for in-memory lookups and is disabled when **Use Disk-Based Lookup** is selected.



When using **Match mode: Regex**, patterns use the JavaScript (ECMAScript) regex engine. If you test patterns in external tools such as Regex101, select the **ECMAScript (JavaScript)** flavor. For more

information, see Regex Flavor.

Reload period (sec): For single-instance deployments, changes to lookup files require an application restart unless a reload interval is enabled. Specify how often (in seconds) Cribl Edge should check for updates and reload the file. Use a positive number to enable automatic reloading. Leave the `-1` (default) to disable. This option is only available for in-memory lookups and is disabled when **Use Disk-Based Lookup** is selected.

In distributed deployments, enabling a **Reload period** can generate conflicts with configuration updates, causing Pipelines to skip executing some Lookup Functions. Cribl recommends that you enable it only for lookup files not managed by Cribl Edge's UI, or lookup files that can be updated by an external process (such as a threat list that you update via a cron job).

For lookup files that **are** managed by Cribl Edge's UI, a distributed Cribl Edge deployment will override this setting as necessary - skipping checks to prevent conflicts that could trigger skipped lookups. These restrictions do **not** apply to single-instance deployments.

Match type: For CIDR and Regex **Match mode**s, this attribute refines how to resolve multiple matches. `First match` will return the first matching entry. `Most specific` will scan all entries, finding the most specific match. `All` will return all matches in the output, as arrays. (Defaults to `First match`. Not displayed for Exact **Match mode**.)

Lookup fields (.csv): Field(s) that should be used to key into the lookup table.

- **Lookup Field Name in Event**: Exact field name as it appears in events. Nested addressing supported.
- **Corresponding Field Name in Lookup**: The field name as it appears in the lookup file. Defaults to the **Lookup field name in event** value. This input is optional.

For disk-based lookups, the field you reference in the Lookup Function must be indexed, or the lookup will fail. Make sure all lookup key fields are properly indexed before deploying. See Add Indexes to a Lookup File for more information.

Case-Sensitive / Multiple Matches

Lookups are case-sensitive by default. (See the **Ignore case** option below.)

If the lookup file contains duplicate key names with different values, all **Match mode**s of this Function will use **only** the value in the key's **final** instance, ignoring all preceding instances.

Output field(s): Field(s) to add to events after matching the lookup table. Defaults to **all** if not specified.

- **Output Field Name from Lookup**: Field name, as it appears in the lookup file.

- **Lookup Field Name in Event**: Field name to add to event. Defaults to the lookup field name. This input is optional. Nested addressing is supported.
- **Default Value**: Optional string value to assign to the field when the lookup entry is not found.

Advanced Settings

Ignore case: Ignore case when performing **Match mode: Exact** lookups. Default is toggled off.

Add to raw event: Whether to append the looked-up values to the `_raw` field, as key=value pairs. Default is toggled off.

Examples

Example 1: Regex Lookups

Assign a `sourcetype` field to events if their `_raw` field matches a particular regex.

paloalto.csv

```
regex,sourcetype
"^[^,]+,[^,]+,[^,]+,THREAT",pan:threat
"^[^,]+,[^,]+,[^,]+,TRAFFIC",pan:traffic
"^[^,]+,[^,]+,[^,]+,SYSTEM",pan:system
```

Copy

Match mode: Regex

Match type: First match



When using the Lookup Function with Regex and `First match`, ensure that your lookup file contains no empty lines - not even at the bottom. Any empty row will cause the function to always return `true`.

Lookup field name in event: `_raw`

Corresponding field name in lookup: `regex`

Events before and after

```
### BEFORE:

{"_raw": "Sep 20 13:03:55 PA-VM 1,2018/09/20 13:03:58,FOOBAR,TRAFFIC,end,2049,2018/09
{"_raw": "Sep 20 13:03:55 PA-VM 1,2018/09/20 13:03:58,FOOBAR,THREAT,end,2049,2018/09/


### AFTER:

{"_raw": "Sep 20 13:03:55 PA-VM 1,2018/09/20 13:03:58,FOOBAR,TRAFFIC,end,2049,2018/09
  "sourcetype": "pan:traffic"
  }
{"_raw": "Sep 20 13:03:55 PA-VM 1,2018/09/20 13:03:58,FOOBAR,THREAT,end,2049,2018/09/
  "sourcetype": "pan:threat"
  }
```

Copy

Example 2: CIDR Lookups

Assign a `location` field to events if their `destination_ip` field matches a particular CIDR range.

paloaltoips.csv

```
range,location
10.0.0.0/24,San Francisco
10.0.0.0/16,California
10.0.0.0/8,US
```

Copy

Match mode: CIDR

Match type: See options below

Lookup field name in event: `destination_ip`

Corresponding field name in lookup: `range`

In **Match mode: CIDR** with **Match type: Most specific**, the lookup will implicitly search for matches from most specific to least specific. There is no need to pre-sort data.

Note that **Match mode: CIDR** with **Match type: First Match** is likely the most performant with large lookups. This can be used as an alternative to **Most specific**, if the file is sorted with the most

specific/relevant entries first. This mode still performs a table scan, top to bottom.

Events before and after

```
### BEFORE:

{"_raw": "Sep 20 13:03:55 PA-VM 1, 2018/09/20 13:03:58,FOOBAR,TRAFFIC,end,2049,2018/0
  "destination_ip": "10.0.0.102"
  }

### AFTER with Match Type: First Match

{"_raw": "Sep 20 13:03:55 PA-VM 1, 2018/09/20 13:03:58,FOOBAR,TRAFFIC,end,2049,2018/0
  "destination_ip": "10.0.0.102",
  "location": "San Francisco"
  }

### AFTER with Match Type: Most Specific

{"_raw": "Sep 20 13:03:55 PA-VM 1, 2018/09/20 13:03:58,FOOBAR,TRAFFIC,end,2049,2018/0
  "destination_ip": "10.0.0.102",
  "location": "San Francisco"
  }

### AFTER with Match Type: All

{"_raw": "Sep 20 13:03:55 PA-VM 1, 2018/09/20 13:03:58,FOOBAR,TRAFFIC,end,2049,2018/0
  "destination_ip": "10.0.0.102",
  "location": [
    "San Francisco",
    "California",
    "US",
  ]}
```

Copy

See Also

- About Lookups
- Configure Lookups

- Lookup Function
- C.Lookup() Expression
- Redis Function for faster lookups using a Redis integration.
- The Cribl Knowledge Pack demonstrates using a Lookup Function to drop Cisco ASA events, or to return a regex for extracting fields.

15.21. Mask

The Mask Function replaces patterns in events, a process called *masking*. Masking is especially useful for redacting PII (personally identifiable information) and other sensitive data.



This Function uses the JavaScript (ECMAScript) regex engine. If you test patterns in external tools such as Regex101, select the **ECMAScript (JavaScript)** flavor. For more information, see Regex Flavor.

Usage

Filter: Filter expression (JavaScript) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Optionally, enter a simple description of this Function.

Final: Toggle on to stop feeding data to the downstream Functions. Default is toggled off.

Masking rules: This table defines pairs of patterns to match and replace. Defaults to empty. Select **Add Rule** to add each rule. Each row of the resulting table provides the following options:

- **Match Regex**: Pattern to replace. Supports capture groups. Use `/g` to replace all matches, like so: `/foo(bar)/g`. You can type in a pattern, or use one of the predefined Regexes.
- **Replace Expression**: A JavaScript expression or literal to replace all matching content. Capture groups can be referenced with `g` and the group number – for example, `g2` to reference the second capture group.
- **Enabled**: Toggle this off to disable matching individual rules, while retaining their configuration in the table. Useful for iterative development and debugging.

Apply to fields: Fields on which to apply the masking rules. Defaults to `_raw`. Add more fields by typing in their names, separated by hard returns. Supports wildcards (`*`) and nested addressing. Supports double-underscore (`__`) internal fields only if individually enumerated - not via wildcards.



Negated terms are supported. When you negate field names, the fields list is order-sensitive. For example, `!foobar` before `foo*` means "Apply to all fields that start with `foo`, except `foobar`." However, `!foo*` before `*` means "Apply to all fields, except for those that start with `foo`."

Advanced Settings

Evaluate fields: Optionally, specify fields to add to events in which one or more of the **Masking rules** were matched. These fields can be useful in downstream processing and reporting. You specify the fields as key-value expression pairs, like those in the Eval Function.

- **Name**: Field name.
- **Value Expression**: JavaScript expression to compute the value (can be a constant).

Depth: The depth to which the Function will traverse nested properties in an event. Defaults to 5.

Evaluating the Replace Expression

The **Replace expression** field accepts a full JS expression that evaluates to a value, so you're not necessarily limited to what's under `C.Mask`. For example, you can do conditional replacement: `g1%2==1 ? `fieldA="odd"` : `fieldA="even"``

The **Replace expression** can reference other event fields as `event.<fieldName>`. For example, `` `${g1}${event.source}` ``. Note that this is slightly different from other expression inputs, where event fields are referenced without `event.` Here, we require the `event.` prefix for the following reasons:

- We don't expect this to be a common case.
- Expanding the event in the replace context would have a high performance hit on the common path.
- There is a slight chance that there might be a `gN` field in the event.



The output of the Mask Function is a string even when the **Replace expression** produces a number.

For example, suppose a **Replace expression** that generates a random integer is applied to a field named `bytes` and produces the number `8403`. Cribl Edge will output an event where the value of the `bytes` field is the **string** `8403`.

If you need to convert the string value back to a number, you could add a Numerify Function to your Pipeline.

Examples

Example 1: Transform a String

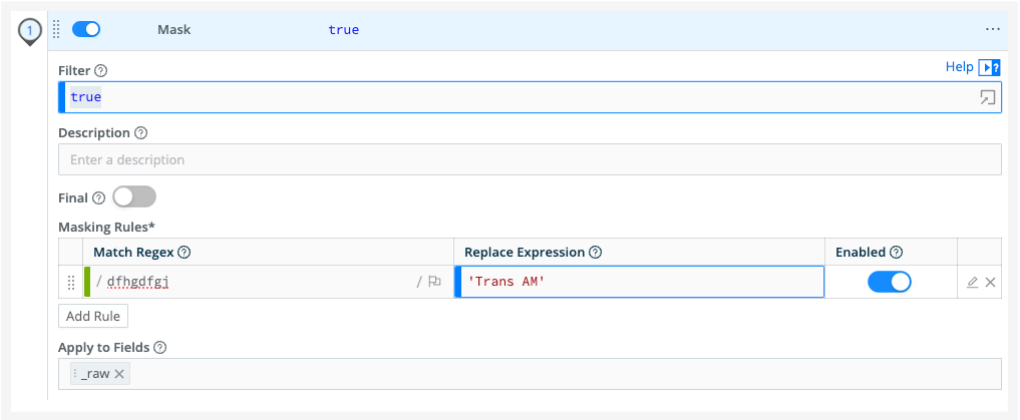
Here, we'll simply search for the string `dfhgdfgj`, and replace that value (if found) with `Trans AM`. This will help close America's muscle-car gap:



Event before masking

Configure the Mask Function > Masking rules as follows:

Match Regex: `dfhgdfgj` Replace Expression: `Trans AM`



Mask Function configuration

Result: Vroom vroom!



Event after masking

Example 2: Mask Sensitive Data

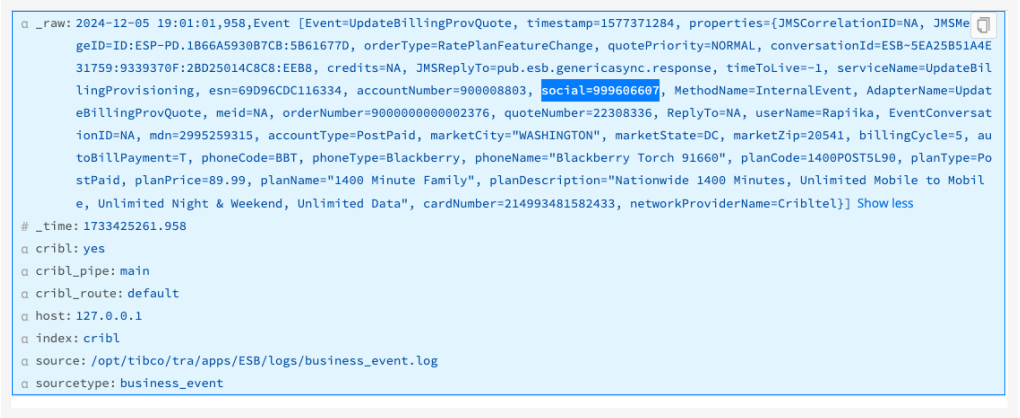
This section demonstrates how to use Mask Functions to anonymize sensitive data, with examples for Social Security numbers and credit card numbers.

In these examples, the Mask Function runs an md5 hash of the values for the specified keys and replaces the original values with the hashed values. Everything in the **Match Regex** field is replaced by the expression defined in the **Replace Expression** field. If you prefer, you can use capture groups in **Match Regex** to define individual string components for manipulation. You can also use string literals in **Replace Expression** to retain static text. Cribl does not retain any content that matches the **Match Regex** that is not inserted into the **Replace Expression**.

If you need to send the unmodified values to a Destination, such as an archival stores, you can use the **Destination** field for the associated Route to narrow the scope of the Mask Function.

Mask Social Security Numbers

This example assumes that you're ingesting data whose `_raw` fields contain unredacted Social Security numbers in the Key=Value pattern `social=#########`.



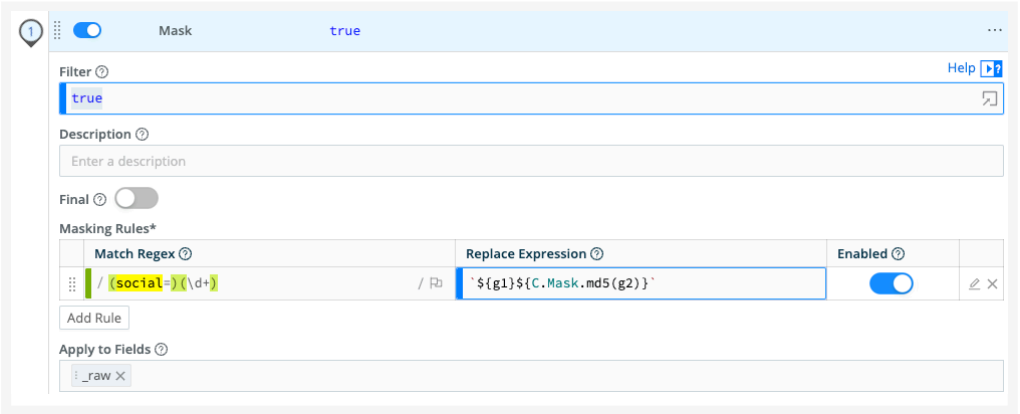
Event with unredacted SSNs

To use a Mask Function to replace the original numeric values for all `social` keys with hashed values, enter the following values into the **Masking rules** fields:

- **Match Regex**: `(social=)(\d+)`
- **Replace Expression**: `` `${g1}${C.Mask.md5(g2)}` ``

Everything in **Match Regex** is replaced by the expression defined in the **Replace Expression** field. `social=` is assigned to capture group `g1` for later reference. The value of `social=` will be hashed because it is referenced as `g2` in the md5 function.

If you don't make `social=` its own capture group (or specify `social=` as a literal in **Replace Expression**) then you cannot reference it using `g1` in **Replace Expression**. The value will be assigned to `g1` along with `social=`, and the entire `social=#########` string will be replaced with a hash of the Social Security number. Without the key name preceding the hashed value, it isn't clear what value is being hashed.



Mask Function configuration for masking Social Security numbers

Result: The sensitive values of `social=` are replaced by their md5 hashes.



Event with hashed Social Security numbers

Mask Credit Card Numbers

This example assumes that you're ingesting data whose `_raw` fields contain unredacted credit card numbers in the values of the `cardNumber` key. The `cardNumber` values may be digits alone or may include hyphens or spaces as separators.



Event with unredacted credit card numbers

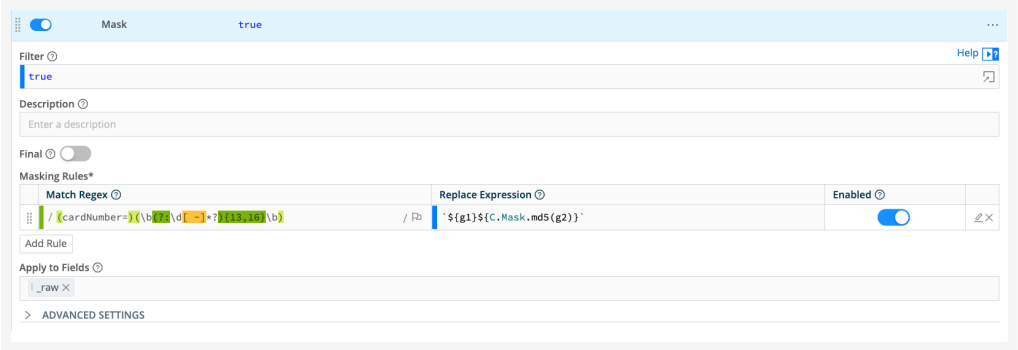
To use a Mask Function to replace the original numeric values for all `cardNumber` keys with hashed values, configure the **Masking rules** fields:

- **Match Regex**: `(cardNumber=)(\b(?:\d[ -]*?){13,16}\b)`
- **Replace Expression**: `` `${g1}${C.Mask.md5(g2)}` ``

Everything in **Match Regex** is replaced by the expression defined in the **Replace Expression** field. `cardNumber=` is assigned to capture group `g1` for later reference. The value of `cardNumber=` will be hashed because it is referenced as `g2` in the md5 function.

If you don't make `cardNumber=` its own capture group (or specify `cardNumber=` as a literal in **Replace Expression**) then you cannot reference it using `g1` in **Replace Expression**. The value will be assigned to `g1`

along with `cardNumber=`, and the entire string, including `cardNumber=`, will be replaced with a hash of the credit card number. Without the key name preceding the hashed value, it isn't clear what value is being hashed.



Mask Function configuration for masking credit card numbers

Result: The sensitive values of `cardNumber=` are replaced by their md5 hashes.



Event with hashed credit card numbers

Example 3: Replace with an Event Field

In this example, we'll replace the IP address `127.0.0.1` in the `_raw` field with the IP address `192.168.123.25` from an existing field named `source_ip`. The following is a snippet of `_raw`:

```
ProcessId=0x0 IpAddress=127.0.0.1 IpPort=0
```

Copy

To match the IP address, we'll define three capture groups. Set **Match Regex** to:

```
/(IpAddress=)((?:\d{1,3}\.){3}\d{1,3})(\s)/
```

Copy

In the **Replace Expression**, we'll reference the `source_ip` field by prepending `event` to it, like this:

```
${g1}${event.source_ip}${g3}
```

Copy

Note that we've also referenced the first and third capture groups, using `g1` and `g3`. This changes `_raw` to:

```
ProcessId=0x0 IpAddress=192.168.123.25 IpPort=0
```

Copy

15.22. Numerify

The Numerify Function converts event fields that are numbers to type `number`.

Usage

Filter: Filter expression (JavaScript) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Simple description about this Function. Defaults to empty.

Final: Toggle on to stop feeding data to the downstream Functions. Default is toggled off.

Depth: Controls how deep the function searches nested events. Higher values may slow performance. Defaults to `5`. Maximum is `10`.

Ignore fields: Specify fields to **not** numerify. Type in field names, separated by hard returns. Supports wildcards (`*`) and nested addressing. When empty (the default), Numerify applies to **all** fields. When populated, takes precedence over the **Include expression**. Also supports negated terms. For example, `!foo*` before `*` means "Ignore all fields, except for those that start with `foo`." See Build Custom Logic to Route and Process Your Data: Wildcard Lists for more information.

Include expression: Optional JavaScript expression to specify fields to numerify. If empty (the default), the Function will attempt to numerify all fields - except those listed in **Ignore fields**, which takes precedence. Use the `name` and `value` global variables to access fields' names/values. (Example: `value != null`.) You can access other fields' values via `__e.<fieldName>`.

Format: Optionally, reformat or truncate the extracted numeric value. Select one of:

- **None**: Applies no reformatting (the default).
- **Floor**: Rounds the number down to the lower adjacent integer (truncates it).
- **Ceil**: Rounds the number up to the higher adjacent integer, removing decimal digits.
- **Round**: Rounds (truncates) the number to a specified number of digits. This option exposes an extra field:
    - **Digits**: Number of digits after the decimal point. Enter a value between `0-20`; defaults to `2`.

Examples

Scenario A:

Assume an event whose text contains a numeric value that must be extracted to perform some numeric analysis. The text looks like this:

```
version=11.5.0.0.1.1588476445
```

We can extract the numeric value by chaining together two Functions:

1. A Regex Extract Function. Set its **Regex** field to `/version=(?<ver>\d+)/`, to capture the first set of digits found in the event string.
2. Then use Numerify.

This captures the substring `11` and converts it to a numeric `11` value.

Scenario B:

Assume email transaction log events like the sample below. The final field is the message's size, in bytes. We want to extract this as a numeric value, for analysis in Cribl Edge or downstream services:

```
03:19 03:22 SMTPD (00180250) [209.221.59.70] C:\IMail\spool\D28de0018025017cd.SMD 3827
```

Again, we can accomplish this with two Functions:

1. A Regex Extract Function. To capture a substring of digits that follows six other substrings (all separated by white space), we set the **Regex** field to: `\S+\s+\S+\s+\S+\s+\S+\s+\S+\s+\S+\s+(?<bytes>\d+)`
2. Then use Numerify.

15.23. OTLP Logs

The OTLP Logs Function ingests logs sent by the Cribl Edge OTel Source, normalizes (that is, formats) and optionally batches them, and sends them on to the Cribl Edge OTel Destination.

How the OTel Source Transforms Log Events

The events that this Function processes can only come from the Cribl Edge OTel Source which itself always ingests events in the form of OTLP logs protocol buffers (protobufs). These protobufs are structures that nest objects within other objects. When you get to the bottom of the nested structure, what you've got is log entries!

Each OTLP logs protobuf contains top-level `ResourceLogs` objects that associate an OTLP Resource with one or more log entries. To get familiar with the internal nomenclature of the OTLP logs protobuf, let's look at its nested objects from the top down.

Object	Description
Log protobuf message	Contains one or more `ResourceLogs`
`ResourceLogs`	Contains one `Resource` and one or more `ScopeLogs`
`ScopeLogs`	Contains one `Scope` and one or more `LogRecords`
`LogRecord`	The log entry

The OTel Source takes these OTLP log protobufs and transforms them into Cribl events that the OTLP Logs Function can ingest.

The OTel Source can send Cribl events in two forms: with or without logs extracted. Which form it sends depends on whether **Advanced Settings** > **Extract logs** is turned on or off in the Source.

When **Extract logs** is off:

- The OTLP Logs Function cannot batch the incoming events, even when **Batch OTLP logs** is turned on. The Function just cleans up the events and sends them on.
- The Cribl events sent to the OTLP Logs Function can include log events, non-log events, or both. The OTLP Function will always send log events through to the OTel Destination. If non-log events arrive, the Function will send them through or drop them, depending on whether its **Drop non-log events** setting is turned on or off.

When **Extract logs** is on:

- The OTLP Logs Function will batch the incoming events if **Batch OTLP logs** is turned on. Otherwise the Function just cleans up the events and sends them on.
- Compared to the original OTLP log protobufs, the Cribl events sent by the OTel Source have a flatter structure: the `resource` and `scope` objects and the extracted records are all peers. The OTLP Logs Function detects this, and if batching is enabled, batches the events such that for each `resource` object there is one `scope_logs` object containing multiple pairs of `scope` and `log_records` objects.

How This Function Batches Logs

For OTLP logs, batching means grouping log events that arrive within a specified time window and whose OTLP Resource Attributes are identical. For example, if the Function receives a thousand events in a period of 30 seconds and those events originate from a set of 100 servers, each event will have a set of Resource Attributes, one or more of which will describe the server where it came from. For simplicity, let's assume that the Resource Attributes of all these events are identical except for the ones that describe the server. The Function will create separate batches for each of those 100 servers because the Resource Attributes differentiate events based on which server it comes from.

Each incoming log event contains both its resource attributes and the log entry itself. In a batch, the resource attributes need only be written once, greatly reducing the amount of data sent over the wire.



For detailed insights into events that passed through the Function but were dropped, set the logging level for channel `func:otlp_logs` to `debug`, and there will be a stats log message output once per minute with the relevant information.

Usage



Ideally, this Function should process events and/or logs **after** other Functions have formatted them. Therefore Cribl recommends that you place this Function last in the Pipeline.

If no metric events appear in Simple or Full Preview, try enabling **Show Dropped Events**: If Cribl Edge processed and dropped events, there's an issue with your data. Either the incoming events are malformed, or, the resource attributes in those events do not match any of the resource attribute prefixes you configured. See **Resource attribute prefixes** below.

Filter: Filter expression (JavaScript) that selects data to feed through the Function. The default `true` setting passes all events through the Function.

Description: Simple description of this Function's purpose in this Pipeline. Defaults to empty.

Final: Toggle on to stop feeding data to the downstream Functions. Default is toggled off.

Drop non-log events: Whether or not to drop any non-log events. Default is toggled off.

Batch OTLP logs: Whether to batch OTLP logs by shared top-level `resource` attributes. The Function only tries to batch events where the logs have been extracted - that is, where the sending OTel Source had **Advanced Settings > Extract logs** enabled when processing the events.

When enabled, exposes these additional controls:

- **Batch size**: The **number of log records** after which a batch will be sent, regardless of whether the timeout duration has been reached.
- **Batch timeout (ms)**: Time duration after which a batch will be sent regardless of size. Defaults to `200` milliseconds.
- **Batch size limit (kb)**: Maximum batch size **in kilobytes**. `0` means no limit.
- **Batch log metadata keys**: Optionally, add metadata keys that you want the Function to create batcher instances for, in addition to any batching based on resource attributes. When set, this processor will create one batcher instance per distinct combination of values in the metadata.
- **Metadata cardinality limit**: Limits the number of unique combinations of metadata key values that the batcher will process. Once the limit has been reached, the Function will drop events whose metadata key values constitute a new combination. Defaults to `1000`.

Basic Example

Filter: `__inputId=='open_telemetry:open_telemetry'`

This will process all events from an OTel Source; in this example we assume that the Source has **Advanced Settings > Extract logs** turned off. Metrics and traces will either be dropped, or emitted without modification, depending on configuration. Unextracted logs will be formatted and emitted; extracted logs will be formatted and either batched or emitted (again depending on configuration).

The Function converts the Cribl event such that the Cribl Edge OTel Destination can ingest it and convert it to an OTLP log event.

See the sample incoming and outgoing events.

Advanced Example

Here, we have two incoming events sent by a OTel Source with **Advanced Settings > Extract logs** turned on.

The Function, with **Batch OTLP logs** turned on, batches the Cribl events such that the Cribl Edge OTel Destination can ingest the batch and convert to a batched OTLP log event.

See the sample incoming and outgoing events.

Sample Events

Basic Example Incoming Event

From OTel Source with extract logs turned off.


```
{
    "resource": {
        "attributes": {
            "telemetry.sdk.language": "python"
        }
    },
    "scope_logs": [{
        "scope": {
            "name": "opentelemetry.sdk._logs._internal",
            "version": "unknown"
        },
        "log_records":[{
            "time_unix_nano": 1717010319837985500,
            "observed_time_unix_nano": 0,
            "body": "Recommendation service started, listening on port 9001",
            "dropped_attributes_count": 0,
            "severity_number": 9,
            "severity_text": "INFO"
        }]
    }],
    "_time": 1717010325.183
}
```


Copy

Basic Example Outgoing Event

When the OTLP Function processes the incoming event above, the event below is the result. It does not matter whether **Batch OTLP logs** is turned on or off - the Function will not try to batch because in the incoming event, logs were not extracted.

```
{
    "resource": {
        "attributes": [{
            "key": "telemetry.sdk.language",
            "value": {
                "string_value": "python"
            }
        }]
    },
    "scope_logs": [{
        "scope": {
            "name": "opentelemetry.sdk._logs._internal",
            "version": "unknown",
            "attributes": []
        },
        "log_records":[{
            "time_unix_nano": 1717010319837985500,
            "observed_time_unix_nano": 0,
            "body": { "string_value": "Recommendation service started, listening on p
            "attributes": [],
            "dropped_attributes_count": 0,
            "severity_number": 9,
            "severity_text": "INFO"
        }]
    }],
    "_time": 1717010325.183
}
```


Copy

Advanced Example Incoming Events

In order to show batching in the outgoing events, let's use two incoming events from an OTel Source where **Advanced Settings** > **Extract logs** is turned on.

Incoming Sample Event 1

```
{
    "schema_url": "https://opentelemetry.io/schemas/1.23.1",
    "resource": {
        "attributes": {
            "telemetry.sdk.language": "python"
        }
    },
    "scope": {
        "name": "kafka.server.KafkaRaftServer",
        "version": "unknown"
    },
    "time_unix_nano": 1717016223073301200,
    "observed_time_unix_nano": 1717016223073302800,
    "body": "[KafkaRaftServer nodeId=1] Kafka Server started",
    "dropped_attributes_count": 0,
    "severity_number": 9,
    "severity_text": "INFO",
    "_time": 1717016223.0733013
}
```


Copy

Incoming Sample Event 2

```
{
    "schema_url": "https://opentelemetry.io/schemas/1.23.1",
    "resource": {
        "attributes": {
            "telemetry.sdk.language": "python"
        }
    },
    "scope": {
        "name": "kafka.server.KafkaBoatServer",
        "version": "unknown"
    },
    "time_unix_nano": 1717016223073401200,
    "observed_time_unix_nano": 1717016223073402800,
    "body": "[KafkaBoatServer nodeId=1] Boat Server started",
    "dropped_attributes_count": 0,
    "severity_number": 9,
    "severity_text": "INFO",
    "_time": 1717016224.0733013
}
```


Copy

Advanced Example Outgoing Batched Events

When the OTLP Function processes the incoming events above, the event below is the result. Here, **Batch OTLP logs** is turned on - the Function is able to batch because in the incoming events, logs were extracted.

```
{
    "resource": {
        "attributes": [{
            "key": "telemetry.sdk.language",
            "value": {
                "string_value": "python"
            }
        }]
    },
    "scope_logs": [
        {
            "scope": {
                "name": "kafka.server.KafkaRaftServer",
                "version": "unknown",
                "attributes": []
```

15.24. OTLP Metrics

The OTLP Metrics Function transforms dimensional metrics events into the OTLP (OpenTelemetry Protocol) format. This Function can be used to send metric events (events that contain the internal field `__criblMetrics`) to any OpenTelemetry Metrics-capable destinations.

For detailed insights into events that passed through the Function but were dropped, set the logging level for channel `func:otlp_metrics` to `debug`, and there will be a stats log message output once per minute with the relevant information.

This Function will delete the internal attribute `__criblMetrics`, making events only suitable for destinations that use the OTLP format.

Usage

Ideally, this Function should process events and/or metrics **after** other Functions have formatted them. Therefore Cribl recommends that you place this Function last in the Pipeline.

If no metric events appear in Simple or Full Preview, try enabling **Show Dropped Events**: If Cribl Edge processed and dropped events, there's an issue with your data. Either the incoming events are malformed, or, the resource attributes in those events do not match any of the resource attribute prefixes you configured. See **Resource attribute prefixes** below.

Filter: Filter expression (JavaScript) that selects data to feed through the Function. The default `true` setting passes all events through the Function.

Description: Simple description of this Function's purpose in this Pipeline. Defaults to empty.

Final: Toggle on to stop feeding data to the downstream Functions. Default is toggled off.

Resource attribute prefixes: The prefixes of top-level attributes to add as resource attributes. See the OpenTelemetry documentation for Resource Semantic Conventions.

- Use periods and underscores for precise matching. For example, `process.path`.
- Remove all the prefixes if you want to bring none of the resources over.
- Use the Eval Function to copy nested attributes to the top level for matching. For example, configure **Evaluate Fields** as follows to create a new top-level field named `process_path` with the value copied

from the nested attribute `process.path`. This new field can then be matched using the OTLP Metrics Function's **Resource attribute prefixes**:

- **Name**: `process_path`
- **Value Expression**: `process.path`

Drop non-metric events: Whether or not to drop any non-metric events.

OTLP version: The drop-down offers `0.10.0` (default) and `1.3.1`.

Batch OTLP metrics: Whether to batch OTLP metrics by shared top-level `resource` attributes. When enabled, exposes these additional controls:

- **Batch size**: The number of metric data points after which a batch will be sent, regardless of whether the timeout duration has been reached.
- **Batch timeout (ms)**: Time duration after which a batch will be sent regardless of size.
- **Batch size limit (kb)**: Maximum batch size. `0` means no limit.
- **Batch metrics metadata keys**: Optionally, add metadata keys that you want the Function to create batcher instances for, in addition to any batching based on resource attributes. When set, this processor will create one batcher instance per distinct combination of values in the metadata.
- **Metadata cardinality limit**: Limits the number of unique combinations of metadata key values that the batcher will process. Once the limit has been reached, the Function will drop events whose metadata key values constitute a new combination.

Choosing or Combining OTLP Versions

Because this Function resides in between Cribl Edge OTel Sources and OTel Destinations, each of which can be independently configured to use the same or different OTLP versions, it's important to know the effects of different choices and combinations.

- Cribl Edge supports starting with OTLP 0.10.0 events and converting them to OTLP 1.3.1 events, but does not support the reverse (1.3.1 to 0.10.0).
- When **OTLP version** is set to `1.3.1`, Cribl Edge will attempt to convert metrics or logs formatted for any previous version to 1.3.1. Since beginning with version 1.0.0, OTLP has introduced no new breaking changes, you can use the Function configured for OTLP 1.3.1 as an "OTLP version 0.x to version 1" converter. Most downstream OTLP receivers comply with OTLP 1.0.0 at minimum; but you should verify this for whatever downstream receiver you use.

Basic Example

Filter: `__inputId=='prometheus_rw:prom_rw_in'`

This will grab all events from a Prometheus Remote Write Source and convert the metrics to OTLP format using only default settings in the Function.



Ensure that the internal attribute `__criblMetrics` does not contain `null` or blank values in any of the child object arrays.

See the sample incoming and outgoing events.

Advanced Example

Here, we have two incoming events sent by a OTel Source with **Advanced Settings** > **Extract Metrics** turned on.

The Function, with **Batch OTLP Metrics** turned on, batches the Cribl events such that the Cribl Edge OTel Destination can ingest the batch and convert to a batched OTLP metric event.

See the sample incoming and outgoing events.

Sample Events

Basic Example Incoming Event

Here is an example incoming event.

```
{
  "__criblEventType": "event",
  "__ctrlFields": [],
  "__final": false,
  "__cloneCount": 0,
  "_time": 1700236380.37,
  "_value": 123.456,
  "_metric": "cribl_logstream_health_outputs",
  "_metric_type": "gauge",
  "cribl_host": "P4MD6R-Pv8x",
  "cribl_wp": "w0",
  "event_host": "P4MD6R-Pv8x",
  "event_source": "cribl",
  "host": "P4MD6R-Pv8x",
  "output": "open_telemetry:otel_webhook_dest",
  "source": "cribl",
  "__criblMetrics": [
    {
      "nameExpr": ["_metric"],
      "values": ["_value"],
      "types": ["gauge"],
      "dims": [
        "cribl_host",
        "cribl_wp",
        "event_host",
        "event_source",
        "host",
        "output",
        "source"
      ]
    }
  ],
  "__offset": 8099,
  "__bytes": 228,
  "__srcIpPort": "127.0.0.1:50868",
  "__inputId": "prometheus_rw:prom_rw_in",
  "service.name": "serviceName",
  "service.instanceId": "serviceInstanceId",
  "system": "system",
  "system.name": "systemName",
  "telemetry.sdk.name": "opentelemetry",
  "telemetry.sdk.language": "java",
```

```
  "k8s.cluster.name": "opentelemetry-cluster",
  "cloud.account.id": 1111111,
  "cloud.platform": "aws"
}
```

Copy

Basic Example Outgoing Event

Here is the above sample event after conversion by the OTLP Metrics Function.

```
{
  "_time": 1700236380.37,
  "resource": {
    "attributes": [
      {
        "key": "host",
        "value": {
          "string_value": "P4MD6R-Pv8x"
        }
      },
      {
        "key": "service.name",
        "value": {
          "string_value": "serviceName"
        }
      },
      {
        "key": "service.instanceId",
        "value": {
          "string_value": "serviceInstanceId"
        }
      },
      {
        "key": "system",
        "value": {
          "string_value": "system"
        }
      },
      {
        "key": "system.name",
        "value": {
          "string_value": "systemName"
        }
      },
      {
        "key": "telemetry.sdk.name",
        "value": {
          "string_value": "opentelemetry"
        }
      },
      {
        "key": "telemetry.sdk.language",
```

```
      "value": {
        "string_value": "java"
      }
    },
    {
      "key": "k8s.cluster.name",
      "value": {
        "string_value": "opentelemetry-cluster"
      }
    },
    {
      "key": "cloud.account.id",
      "value": {
        "int_value": 1111111
      }
    },
    {
      "key": "cloud.platform",
      "value": {
        "string_value": "aws"
      }
    }
  ]
},
"instrumentation_library_metrics": [
  {
    "schema_url": "",
    "metrics": [
      {
        "name": "cribl_logstream_health_outputs",
        "gauge": {
          "data_points": [
            {
              "as_double": 123.456,
              "time_unix_nano": 1700236380370000000,
              "start_time_unix_nano": 1700236380370000000,
              "attributes": [
                {
                  "key": "cribl_host",
                  "value": {
                    "string_value": "P4MD6R-Pv8x"
                  }
                },
```

```
            {
              "key": "cribl_wp",
              "value": {
                "string_value": "w0"
              }
            },
            {
              "key": "event_host",
              "value": {
                "string_value": "P4MD6R-Pv8x"
              }
            },
            {
              "key": "event_source",
              "value": {
                "string_value": "cribl"
              }
            },
            {
              "key": "host",
              "value": {
                "string_value": "P4MD6R-Pv8x"
              }
            },
            {
              "key": "output",
              "value": {
                "string_value": "open_telemetry:otel_webhook_dest"
              }
            },
            {
              "key": "source",
              "value": {
                "string_value": "cribl"
              }
            }
          ]
        }
      ]
    },
    "__type": "gauge"
  }
]
```

```
      }
    ]
  }
```

Copy

Advanced Example Incoming Events

In order to show batching in the outgoing events, let's use two incoming events from an OTel Source where **Advanced Settings** > **Extract metrics** is turned on.

Incoming Sample Event 1 - Gauge Metric

```
{
    "_time": 1700236380.37,
    "resource":
    {
        "attributes":
        [
            {
                "key": "service.name",
                "value":
                {
                    "string_value": "serviceName"
                }
            },
            {
                "key": "service.instanceId",
                "value":
                {
                    "string_value": "serviceInstanceId"
                }
            },
            {
                "key": "system.name",
                "value":
                {
                    "string_value": "systemName"
                }
            },
            {
                "key": "telemetry.sdk.name",
                "value":
                {
                    "string_value": "opentelemetry"
                }
            },
            {
                "key": "telemetry.sdk.language",
                "value":
                {
                    "string_value": "java"
                }
            },
            {
```

```
            "key": "k8s.cluster.name",
            "value":
            {
                "string_value": "opentelemetry-cluster"
            }
        }
    ]
},
"instrumentation_library_metrics":
[
    {
        "schema_url": "",
        "metrics":
        [
            {
                "name": "cribl_logstream_health_outputs",
                "gauge":
                {
                    "data_points":
                    [
                        {
                            "as_double": 123.456,
                            "time_unix_nano": 1700236380370000000,
                            "start_time_unix_nano": 1700236380370000000,
                            "attributes":
                            [
                                {
                                    "key": "cribl_host",
                                    "value":
                                    {
                                        "string_value": "P4MD6R-Pv8x"
                                    }
                                },
                                {
                                    "key": "cribl_wp",
                                    "value":
                                    {
                                        "string_value": "w0"
                                    }
                                },
                                {
                                    "key": "event_host",
                                    "value":
```

```
                                {
                                    "string_value": "P4MD6R-Pv8x"
                                }
                            },
                            {
                                "key": "event_source",
                                "value":
                                {
                                    "string_value": "cribl"
                                }
                            },
                            {
                                "key": "host",
                                "value":
                                {
                                    "string_value": "P4MD6R-Pv8x"
                                }
                            },
                            {
                                "key": "output",
                                "value":
                                {
                                    "string_value": "open_telemetry:otel_webh
                                }
                            },
                            {
                                "key": "source",
                                "value":
                                {
                                    "string_value": "cribl"
                                }
                            }
                        ]
                    }
                ]
            },
            "__type": "gauge",
            "description": "This is a metric description"
        }
    ],
    "instrumentation_library":
    {
        "name": "io.opentelemetry.sdk.trace",
```

```
                "version": "1.33.0-alpha"
            }
        }
    ],
    "__outLen": 1266,
    "__bufToken": "otlp-metrics-batcher1"
}
```


Copy

Incoming Sample Event 2 - Summary Metric

```
{
    "__criblEventType": "event",
    "__ctrlFields":
    [],
    "__final": false,
    "__cloneCount": 0,
    "__bytes": 404,
    "__criblMetrics":
    [
        {
            "dims":
            [
                "cribl_host",
                "cribl_wp"
            ],
            "nameExpr":
            [
                "_metric"
            ],
            "types":
            [
                "summary"
            ],
            "values":
            [
                "_value"
            ]
        }
    ],
    "__inputId": "prometheus_rw:prom_rw_in",
    "__offset": 5083,
    "__srcIpPort": "127.0.0.1:59491",
    "_metric": "request_duration_summary",
    "_metric_type": "summary",
    "_time": 1701793930,
    "description": "This is an event description",
    "_value": 0,
    "cribl_host": "P4MD6R-Pv8x",
    "cribl_wp": "w0",
    "request_duration_summary_data":
    {
        "_count": 0,
```

```
        "_quantiles":
        {
            "0.5": 0,
            "0.9": 0
        },
        "_sum": 0
    },
    "instrumentation_library":
    {
        "name": "io.opentelemetry.sdk.trace",
        "version": "1.33.0-alpha"
    },
    "service.name": "serviceName",
    "service.instanceId": "serviceInstanceId",
    "system": "system",
    "system.name": "systemName",
    "telemetry.sdk.name": "opentelemetry",
    "telemetry.sdk.language": "java",
    "k8s.cluster.name": "opentelemetry-cluster",
    "cloud.account.id": 1111111,
    "cloud.platform": "aws"
}
```

Copy

Advanced Example Outgoing Batched Events

Here is the resulting converted OTLP metric events in a single batch.

```
{
    "_time": 1700236380.37,
    "resource":
    {
        "attributes":
        [
            {
                "key": "service.name",
                "value":
                {
                    "string_value": "serviceName"
                }
            },
            {
                "key": "service.instanceId",
                "value":
                {
                    "string_value": "serviceInstanceId"
                }
            },
            {
                "key": "system.name",
                "value":
                {
                    "string_value": "systemName"
                }
            },
            {
                "key": "telemetry.sdk.name",
                "value":
                {
                    "string_value": "opentelemetry"
                }
            },
            {
                "key": "telemetry.sdk.language",
                "value":
                {
                    "string_value": "java"
                }
            },
            {
```

```
            "key": "k8s.cluster.name",
            "value":
            {
                "string_value": "opentelemetry-cluster"
            }
        }
    ]
},
"instrumentation_library_metrics":
[
    {
        "schema_url": "",
        "metrics":
        [
            {
                "name": "cribl_logstream_health_outputs",
                "gauge":
```

15.25. OTLP Traces

The OTLP Traces Function ingests traces sent by the Cribl Edge OTel Source, normalizes (that is, formats) and optionally batches them, and sends them on to Cribl Edge Destinations.

How the OTel Source Transforms Trace Events

This Function can process only events coming from the Cribl Edge OTel Source, which always ingests events in the form of OTLP trace protocol buffers (protobufs). These protobufs are structures that nest objects within other objects. Each OTLP trace protobuf contains top-level `ResourceSpans` objects that associate an OTLP Resource with one or more spans.

Object	Description
Trace protobuf message	Contains one or more `ResourceSpans`
`ResourceSpans`	Contains one Resource and one or more `ScopeSpans`
`ScopeSpans`	Contains one Scope and one or more `Span`
`Span`	The trace entry

How This Function Batches Traces

For OTLP traces, batching means grouping extracted spans that arrive within a specified time window and whose OTLP Resource Attributes are identical. The Function will create separate batches for each unique combination of resource attributes and metadata keys.

Usage



Ideally, this Function should process events and/or traces **after** other Functions have formatted them. Therefore, Cribl recommends that you place this Function last in the Pipeline.

Filter: Filter expression (JavaScript) that selects data to feed through the Function. The default `true` setting passes all events through the Function.

Description: Simple description of this Function's purpose in this Pipeline. Defaults to empty.

Final: If toggled to **Yes**, stops feeding data to the downstream Functions. Defaults to **No**.

Drop non-trace events: Whether or not to drop any non-trace events. Defaults to **No**.

OTLP version: Specifies the version of the OTLP protocol used for formatting and sending output events downstream.

- Version `0.10.0` (Default): This version uses the older OTLP format. It is recommended for users who need compatibility with legacy systems.
- Version `1.3.1`: This version supports newer OTLP features, including `scope_spans` instead of `instrumentation_library_spans`, and additional fields like `flags` in `Link` and `Span` objects. It is recommended for users who want to leverage the latest OTLP features and improvements.

Batch OTLP traces: When set to **Yes**, the Function batches together extracted spans that have the same resource and (optionally) metadata keys until they reach a flush criteria. When set to **No**, the Function will emit extracted/unextracted span events that have been normalized, but they will not be combined with other spans - each event passed to the Function will result in one event being returned.

When set to **Yes**, exposes these additional controls:

- **Batch size**: The number of spans after which a batch will be sent, regardless of whether the timeout duration has been reached.
- **Batch timeout (ms)**: Time duration after which a batch will be sent regardless of size. Defaults to `200` milliseconds.
- **Batch size limit (kb)**: Maximum batch size in kilobytes. `0` means no limit.
- **Batch trace metadata keys**: Optionally, add metadata keys that you want the Function to create batcher instances for, in addition to any batching based on resource attributes. When set, this processor will create one batcher instance per distinct combination of values in the metadata.
- **Metadata cardinality limit**: Limits the number of unique combinations of metadata key values that the batcher will process. Once this limit is reached, events with new combinations of metadata key values will be dropped. Defaults to `1000`.

Example

This example demonstrates how the OTLP Traces Function processes and batches trace events when the **Batch OTLP traces** setting is enabled. The settings used in this example are:

- **Batch OTLP traces**: **Yes**
- **Batch size**: `2`
- **Batch timeout (ms)**: `200`
- **Batch size limit (kb)**: `0` (no limit)
- **Batch trace metadata keys**: **None**
- **Metadata cardinality limit**: `1000`

- Drop non-trace events: Yes

Incoming Event 1


```
{
  "resource": {
    "attributes": {
      "service.name": "currencyservice",
      "service.instance.id": "instance-1"
    }
  },
  "scope": {
    "name": "currencyservice",
    "version": "1.0.0"
  },
  "spans": [{
    "trace_id": "4bf92f3577b34da6a3ce929d0e0e4736",
    "span_id": "00f067aa0ba902b7",
    "name": "GetSupportedCurrencies",
    "start_time_unix_nano": 1616587310000000000,
    "end_time_unix_nano": 1616587311000000000,
    "attributes": {
      "http.method": "GET",
      "http.url": "/currencies"
    }
  }]
}
```


Copy

Incoming Event 2

```
{
  "resource": {
    "attributes": {
      "service.name": "currencyservice",
      "service.instance.id": "instance-1"
    }
  },
  "scope": {
    "name": "currencyservice",
    "version": "1.0.0"
  },
  "spans": [{
    "trace_id": "4bf92f3577b34da6a3ce929d0e0e4736",
    "span_id": "00f067aa0ba902b8",
    "name": "ConvertCurrency",
    "start_time_unix_nano": 1616587312000000000,
    "end_time_unix_nano": 1616587313000000000,
    "attributes": {
      "http.method": "POST",
      "http.url": "/convert"
    }
  }]
}
```


Copy

Outgoing Batched Event

```
{
  "resource": {
    "attributes": [
      {
        "key": "service.name",
        "value": {
          "string_value": "currencyservice"
        }
      },
      {
        "key": "service.instance.id",
        "value": {
          "string_value": "instance-1"
        }
      }
    ]
  },
  "scope_spans": [
    {
      "scope": {
        "name": "currencyservice",
        "version": "1.0.0"
      },
      "spans": [
        {
          "trace_id": "4bf92f3577b34da6a3ce929d0e0e4736",
          "span_id": "00f067aa0ba902b7",
          "name": "GetSupportedCurrencies",
          "start_time_unix_nano": 1616587310000000000,
          "end_time_unix_nano": 1616587311000000000,
          "attributes": [
            {
              "key": "http.method",
              "value": {
                "string_value": "GET"
              }
            },
            {
              "key": "http.url",
              "value": {
                "string_value": "/currencies"
              }
            }
```

```
    }
  ]
},
{
  "trace_id": "4bf92f3577b34da6a3ce929d0e0e4736",
  "span_id": "00f067aa0ba902b8",
  "name": "ConvertCurrency",
  "start_time_unix_nano": 1616587312000000000,
  "end_time_unix_nano": 1616587313000000000,
  "attributes": [
    {
      "key": "http.method",
      "value": {
        "string_value": "POST"
      }
    },
```

15.26. Parser

The Parser Function can be used to extract fields out of events or reserialize (rewrite) events with a subset of fields. Reserialization will preserve the format of the events.

For example, if an event contains comma-delimited fields, and `fieldA` and `fieldB` are filtered out, those fields' positions will be set to `null` but not deleted completely.

The Parser Function can't remove fields that it didn't create. A subsequent Eval Function can do this.

Usage

Filter: Filter expression (JavaScript) that selects data to feed through the Function. Defaults to `true`, meaning evaluate all events.

Description: Simple description about this Function. Defaults to empty.

Final: Toggle on to stop feeding data to the downstream Functions. Default is toggled off.

Operation mode: **Extract** will create new fields. **Reserialize** will extract, filter fields, and then reserialize.

Type: Parser/Formatter type to use. Options:

- **CSV**: Comma-separated values.
- **Extended Log File Format**: Extended Log File Format.
- **Common Log Format**: Common Log Format.
- **Key=Value Pairs**: A set of data that represents two associated groups through a key and a value.
- **JSON Object**: JavaScript Object Notation (JSON) is a standard text-based format for representing structured data based on JavaScript object syntax.
- **Delimited values**: A character identifies the beginning or the end of a character string.
- **Regular Expression**: A sequence of characters that specifies a match pattern in text.
- **Grok**: A string of special characters and regular expressions (pattern) that match data.

The following two additional options appear in the drop-down, but can only be used when **Operation mode** is set to **Extract**; both expand the UI:

- **Grok**
- **Regular Expression**

Library: Select an option from the collection of available Parsers. Although specifying a Library will auto-generate an example list of fields, the list may still need to be modified to accommodate the desired fields from the events as well as the actual field order.

Source field: Field that contains text to be parsed. Not usually needed in Reserialize mode.

Destination field: Name of field in which to place the extracted and serialized fields. If the field already exists, it will be overwritten. If multiple new fields are created and this field is populated, then all new fields are created as elements of an array. The array's name will be set to this field's value. If you want all new fields to be independent, rather than in an array, then specify them using **List of fields**, below. (Extract and Reserialize modes only.)

Clean fields: This option appears for **Type: K=V Pairs**. Toggle on to clean field names by replacing non-alphanumeric characters with `_`. This will also strip leading and trailing `"` symbols.

List of fields: Fields expected to be extracted, in order. If not specified, Parser will auto-generate fields.



List of Fields will not display, when the Parser/Formatter **Type** is set to `JSON Object` or `Key=Value Pairs`. This option is only relevant for structured formats like `CSV`, `CLF`, and `ELFF`. It helps name extracted fields when they aren't explicitly defined in the raw data. For `JSON Object` and `Key-Value Pair` formats, field names are inherent and this option is ignored.

Fields to keep: List of fields to keep. When you specify any fields here, Parser keeps only those fields and drops all other parsed fields, even if **Fields to remove** is empty. **Fields to keep** takes precedence over **Fields to remove** and **Fields filter expression**. The field accepts wildcards (`*`), which supports both:

- Wildcards (`*`) to include multiple fields.
- Nested addressing, where you can reference a parent field and all its children using a wildcard. For example, if you convert `_raw` into an object, use `_raw*` to refer to `_raw` and all nested fields inside it. See Wildcard Lists for more information.

Fields to remove: List of fields to remove. Supports wildcards (`*`). Cannot remove fields matching **Fields to keep**. Supports wildcards (`*`) to include multiple fields and nested addressing. See Wildcard Lists for more information.

Fields filter expression: Expression to evaluate against each field's `{index, name, value}` context. Return truthy to keep fields, or falsy to remove fields. Index is zero-based. See details below.

Types That Expand the UI

Setting **Type** to **Delimited Values** displays the following extra options:

- **Delimiter**: Delimiter character to split value. Defaults to comma (`,`). You can also specify pipe (`|`) or tab characters.

- **Quote char**: Character used to quote literal values. Defaults to `"`.
- **Escape char**: Character used to escape the delimiter character. Defaults to: `\`
- **Null value**: Field value representing the null value. These fields will be omitted. Defaults to: `-`

Setting **Type** to **Grok** displays the Grok Function UI; setting it to **Regex Extract** displays the Regex Extract UI. These Parser preserve all previewing and editing. Also, **Grok** enables you to bring Grok Patterns into Cribl Search.

Advanced Settings

Allowed key characters: Enter characters permitted in a key name, even though they are normally separator or control characters. Separate entries with a tab or hard return. This setting does not affect how the value is parsed.

Allowed value characters: Enter characters permitted in a value name, even though they are normally separator or control characters. Separate entries with a tab or hard return. This setting does not affect how the key is parsed.

Fields/Values Interaction

This Function's **Filter**, **Type**, **Fields to keep**, **Fields to remove**, and **Fields filter expression** settings interact as follows.

Order of Evaluation

The order is: **Fields to keep** > **Fields to remove** > **Fields filter expression**.



As soon as you list any fields in **Fields to keep**, Parser treats that list as the complete set of parsed fields to retain. Any parsed field not matching **Fields to keep** is dropped, even if **Fields to remove** is empty.

Precedence

If a field is in both **Fields to keep** and **Fields to remove**, **Fields to keep** takes precedence.

If a field is in both **Fields to remove** and **Fields filter expression**, **Fields to remove** takes precedence.

Negation

Both **Fields to remove** and **Fields to keep** support negated terms. When you use negated terms, your fields list is order-sensitive. For example, `!foobar, foo*` means "All fields that start with `foo`, except `foobar`."

However, `!foo*, *` means “All fields, except for those that start with `foo`.”

Characters to Escape

Particular characters need special treatment when parsed when you select the Key=Value Pairs or CSV Parser/Formatter type.

Key-Value Pairs

When a Parser Function’s **Type** is set to `Key=Value Pairs`: Within your data, field values that contain an `=` delimiter **must** be surrounded by quotes (`"..."`), or the Function will return unexpected results. As an example, here is a set of four fields and values, in which two values are quoted to parse predictably:

```
one=1,two="value=2",three=3,four="value=4"
```

These would resolve to the following parsed values:

Field Name	Value
one	1
two	"value=2"
three	3
four	"value=4"

CSV

When a Parser Function’s **Type** is set to `CSV`: Within your data, column values that contain quotes or commas **must** quote those characters, or the Function will return unexpected results. To illustrate this, here is a properly formatted example of four CSV values:

```
"value with ""escaped"" quotes is 1",2,3,"you know, this is 4"
```

The first value includes quotation marks, which are each escaped with surrounding quotes, and the whole value is delimited by outer quotes. The fourth value includes a comma, so that value is delimited by outer quotes. These four columns would cleanly evaluate to:

---

value with “escaped” quotes is 1 2 3 you know, this is 4

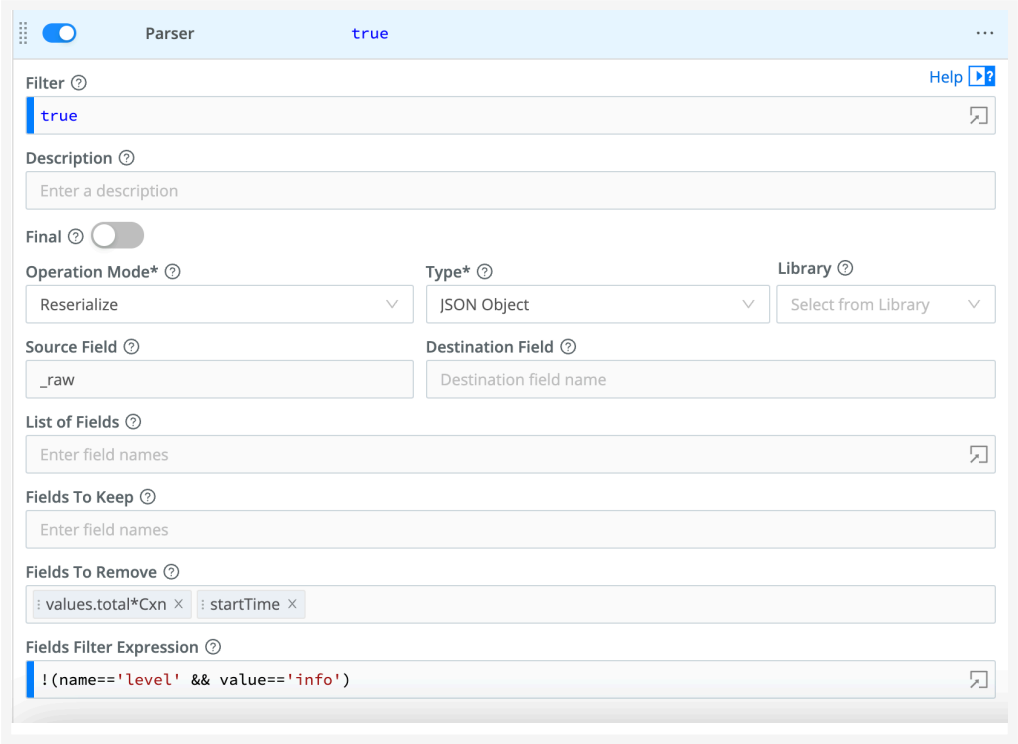
Examples

The following examples show creating and dropping fields, reserialization, and serializing to CSV versus JSON.

Example 1

Insert the following sample, using **Sample Data** > **Import Data**: `2019/06/24 05:10:55 PM Z a=000,b=001,c=002,d=003,e=004,f=005,g1=006,g2=007,g3=008`

Create the following test Parser Function.



Parser Function initial configuration

First, set the **Type** to `Key=Value Pairs`.

Scenario A

Keep fields `a, b, c`. Drop the rest.

This example shows that **Fields to keep** alone is enough to drop unlisted fields. Even though **Fields to remove** is left empty, Parser keeps only the fields listed in **Fields to keep** and drops all other parsed fields.

Expected result: `a, b, c`

- Fields to keep: `a, b, c`

- Fields to remove: [empty]
- Fields filter expression: [empty]

Result: The event will gain four new fields and values, as follows.

- a: `000`
- b: `001`
- c: `002`
- cribl_pipe: `parser2`



Scenario A result

You can check your stats by selecting the **Preview** pane's **Basic Statistics** (chart) button. In the resulting pop-up, the **Number of Fields** should have incremented ty four.

Now that you have the hang of it, try out the other simple scenarios below.

Scenario B

Keep fields `a`, `b`, those that start with `g`. Drop the rest.

Expected result: `a`, `b`, `g1`, `g2`, `g3`

- Fields to keep: `a`, `b`
- Fields to remove: [empty]
- Fields filter expression: `name.startsWith('g')`

Scenario C

Keep fields `a`, `b`, those that start with `g` but only if value is `007`. Drop the rest.

Expected result: `a`, `b`, `g2`

- Fields to keep: `a, b`
- Fields to remove: [empty]
- Fields filter expression: `name.startsWith('g') && value=='007'`

Scenario D

Keep fields `a, b, c`, those that start with `g`, unless it's `g1`. Drop the rest.

Expected result: `a, b, c, g2, g3`

- Fields to keep: `a, b, c`
- Fields to remove: `g1`
- Fields filter expression: `name.startsWith('g')`

Scenario E

Keep fields `a, b, c`, those that start with `g` but only if index is greater than `6`. Drop the rest.

Expected result: `a, b, c, g2, g3`

- Fields to keep: `a, b, c`
- Fields to remove: [empty]
- Fields filter expression: `name.startsWith('g') && index>6`



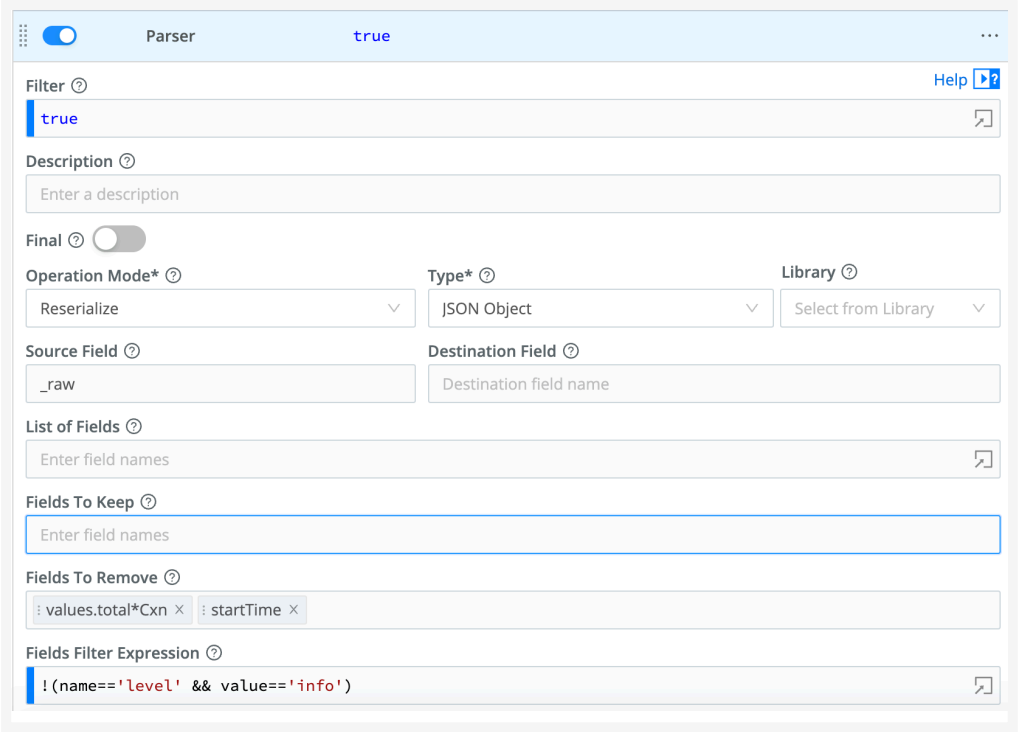
The `index` refers to the location of a field in the array of all fields extracted by **this** Parser. It is zero-based. In the case above, `g2` and `g3` have `index` values of `7` and `8`, respectively.

Example 2

Assume we have a JSON event that needs to be **reserialized**, given these requirements:

1. Remove the `level` field only if it's set to `info`.
2. Remove the `startTime` field, and all fields in the `values.total.` path that end in `Cxn`.

Parser Function configuration:



Parser Function configuration for Example 2

JSON event after being processed by the Function:



Example 2 event transformation

Example 3

Insert the following sample, using **Sample Data** > **Import Data**:

```
2019/06/24 15:25:36 PM Z a=000,b=001,c=002,d=003,e=004,f=005,g1=006,g2=007,g3=008,
```

For all scenarios below, first create a Parser Function to extract all fields, by setting the **Type** to `Key=Value Pairs`. Then add a second Parser Function with the configuration shown under **Parser 2**.

Scenario A

Serialize fields `a, b, c, d` in CSV format.

Expected result: `_raw` field will have this value `000,001,002,003`

Parser 2:

- Operation mode: `Serialize`
- Source field: [empty]
- Destination field: [empty]
- Type: CSV
- List of fields: `a, b, c, d` (needed for positional formats)

Scenario B

Serialize fields `a, b, c` in JSON format, under a field called `bar`.

Expected result: `bar` field will be set to: `{"a":"000","b":"001","c":"002","d":"003"}`

Parser 2:

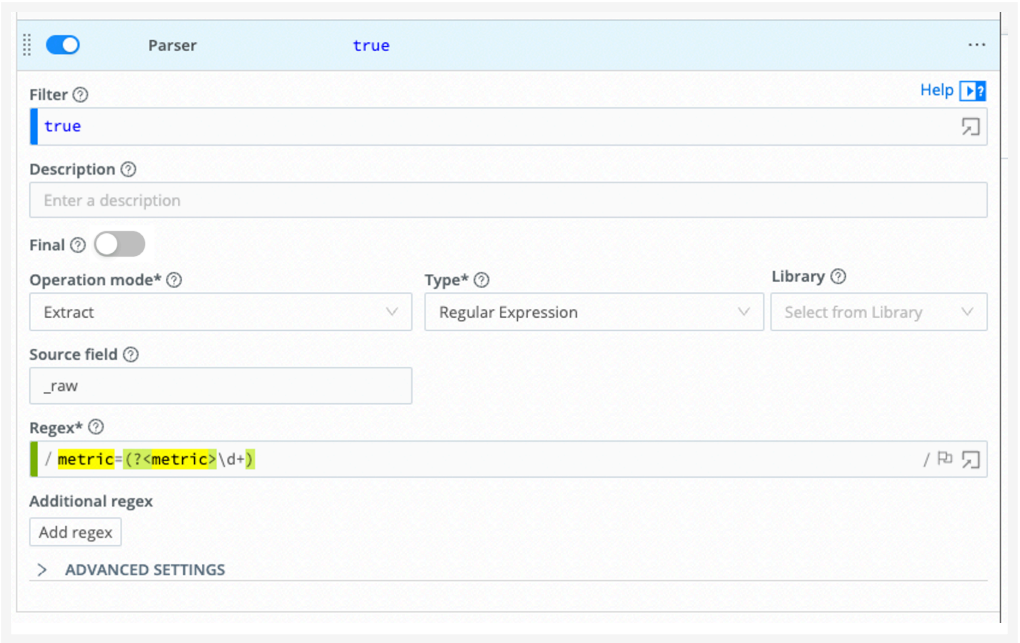
- Operation mode: `Serialize`
- Source field: [empty]
- Destination field: `bar`
- Type: `JSON`
- List of fields: [empty]
- Fields to keep: `a, b, c, d`

Example 4

Assume we have a simple event that looks like this:

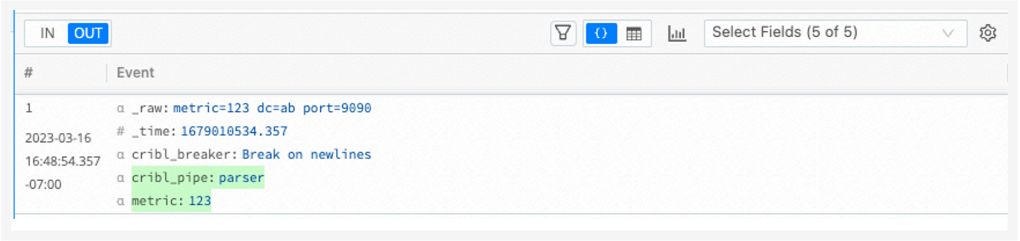
```
metric=123 dc=ab port=9090
```

If we want to extract only the `metric` field, we can configure a Parser of **Type** `Regular Expression`, where the **Regex** field is set to `metric=(?<metric>\\d+)`:



Parser Function configuration for Example 4

The output will look like this:



Parser Function regex output

Example 5

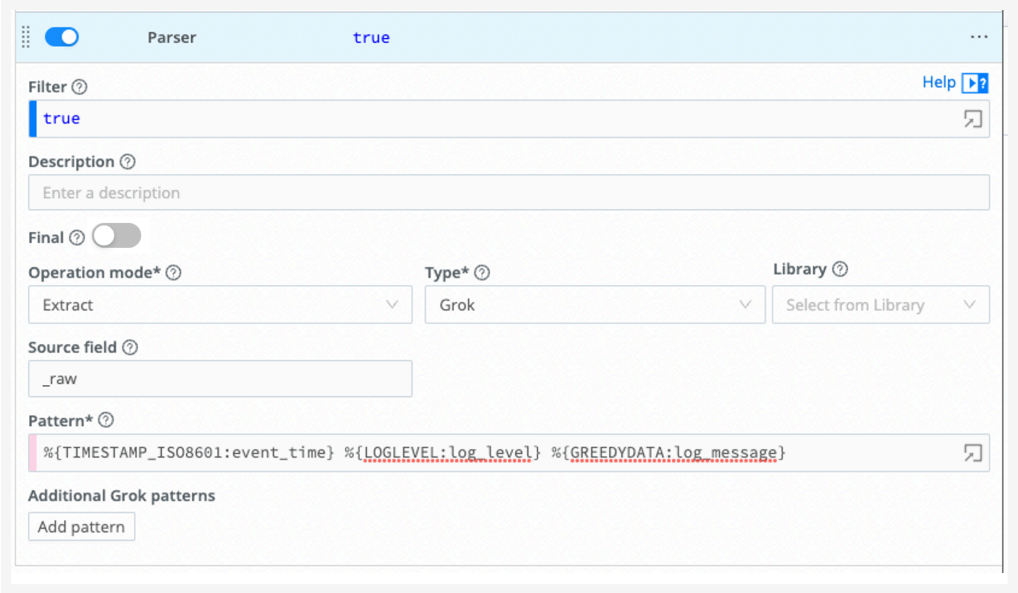
Assume we have an event that looks like this:

```
2020-09-16T04:20:42.45+01:00 DEBUG This is a sample debug log message
```

We can use a Grok pattern to parse the event so that the timestamp, log level, and the message are separated. A Parser of **Type** `Grok` with the following Pattern will work:

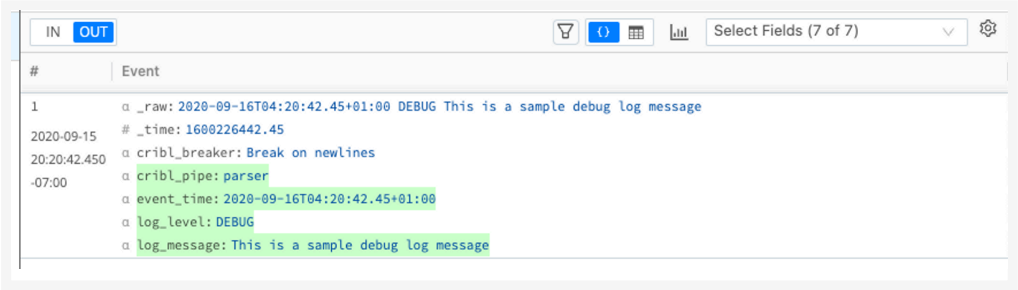
```
%{TIMESTAMP_ISO8601:event_time} %{LOGLEVEL:log_level} %{GREEDYDATA:log_message}
```

The Parser Function configuration looks like this:



Parser Function configuration for Example 5

The output will look like this:

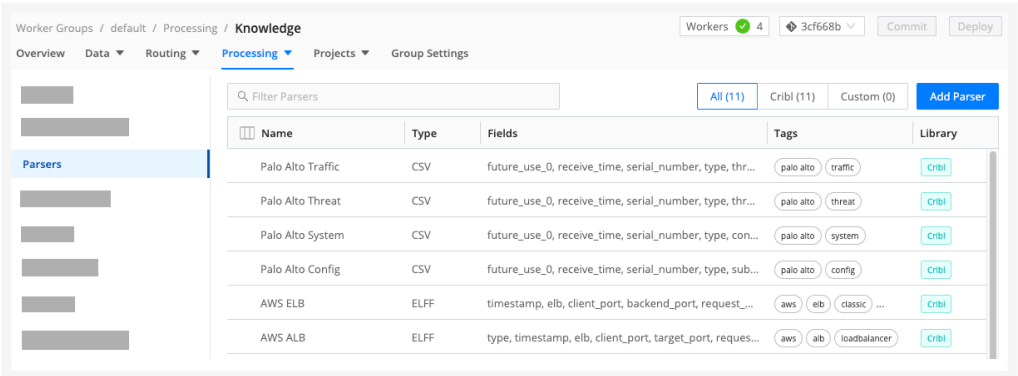


Parser Function grok output

Available Parsers

You can edit existing and create new Parsers for use in the Parser Function. The Parser collection is searchable, and each parser can be tagged as necessary.

To find the collection, select **Fleets** from the sidebar, and choose a Fleet. Then, on the **Fleets** submenu, select **More**, then **Knowledge**, then **Parsers**.



You can refer to a single Parser in multiple Parser Functions in different Pipelines. When editing them, be aware that any change will affect all places it is used in.

Supported Parser Types

- **CSV**: Comma-separated values.
- **Extended Log File Format**: Extended Log File Format.
- **Common Log Format**: Common Log Format.
- **Key=Value Pairs**: A set of data that represents two associated groups through a key and a value.
- **JSON Object**: JavaScript Object Notation (JSON) is a standard text-based format for representing structured data based on JavaScript object syntax.
- **Delimited values**: A character identifies the beginning or the end of a character string.
- **Regular Expression**: A sequence of characters that specifies a match pattern in text.
- **Grok**: A string of special characters and regular expressions (pattern) that match data.

Create a Parser

To create a parser, follow these steps:

1. In the sidebar, select **Fleets** from the sidebar, and choose a Worker Group or Fleet.
2. On the **Fleets** submenu, select **More**, then **Knowledge**, then **Parsers**.
3. Select **Add Parser**.
4. Enter a unique **ID**.
5. Optionally, enter a **Description**.
6. Select a **Type** (see the supported types above).
7. Enter the **List of fields** expected to be extracted, in order. Select this field's Maximize icon (far right) if you'd like to open a modal where you can work with sample data and iterate on results.

15.27. Publish Metrics

The Publish Metrics Function extracts, formats, and outputs metrics from events.

Usage

Filter: Filter expression (JavaScript) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Simple description about this Function. Defaults to empty.

Final: Toggle on to stop feeding data to the downstream Functions. Default is toggled off.

Overwrite: If toggled on, overwrite previous metric specs. Otherwise, append. Default is toggled off.

Metrics

Add metrics: List of metrics to extract from the event and format. Destinations can pass the formatted metrics to a metrics aggregation platform. Select **Add metrics** to add new rows containing the following options:

- **Event Field Name**: The name of the field (in the event) that contains the metric value. Should contain only letters, numbers, underscores (`_`), and `.` characters (to separate names in nested structures).
- **Metric Name Expression**: JavaScript expression to evaluate the metric field name. Defaults to the **Event field name** value.



The JavaScript expression will evaluate the metric field name only after the metrics are processed for transport to the Destination. While in the processing Pipeline, the metric name expression appears as a literal.

To have metric names evaluated in preview, enable the Enhance Metric Output setting.

- **Metric Type**: Select `Gauge` (the default), `Counter`, `Timer`, `Distribution`, `Summary`, or `Histogram`. General definitions that can vary across senders:
  - `Gauge`: A numeric value that can increase or decrease over time - like a temperature or pressure gauge.
  - `Counter`: A cumulative numeric value - it can only increase over time.
  - `Timer`: Generally measures how long a given event type takes (duration), and how often it occurs (frequency).
  - `Distribution`: The statistical distribution of a set of values over a time interval. (This type generally provides raw data, not an aggregation.)

- `Summary`: Tracks the count, sum, and average of the values. Optionally, also tracks quantiles across the values.
- `Histogram`: Tracks the count, sum, and average of the values. Also groups the observations in their corresponding intervals, or buckets.

Remove metrics: Optionally, enter a List of field names to look for when removing metrics. Where a metric's field name matches an element in this list, Cribl Edge will remove that metric from the event.

Dimensions

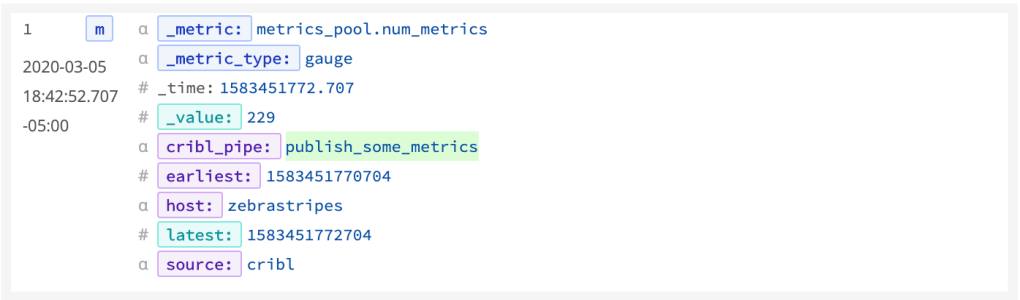
Add dimensions: Optional list of dimensions to include in events. Supports wildcards. If you don't specify metrics, values will be appended to every metric found in the event. When you add a new metric, dimensions will be present only in those new metrics. Defaults to `!_* *`.

Remove dimensions: Optional list of dimensions to associate with every extracted metric value. Leave blank if this Function processes output from the Aggregations Function as dimensions will be automatically discovered.

> The **Add dimensions** and **Remove dimensions** fields support wildcards and negated terms. When you use negated terms, the list is order-sensitive. For example, `!foobar` before `foo*` means "All fields that start with `foo`, except `foobar`." However, `!foo*` before `*` means "All fields, except for those that start with `foo`."
>
> Cribl Edge can send out multi metrics, but this must be configured on compatible Destinations.

Fields Color Coding

On the right Preview pane's **OUT** tab, the Publish Metrics Function adds the following color codes to field labels:



Dimension: purple | **Value**: cyan (light blue) | **Info**: dark blue

These are in addition to the color codes applied to field values, which are listed here.

Examples

Scenario A:

Assume we're working with AWS VPC Flowlog events that have the following structure:

```
version account_id interface_id srcaddr dstaddr srcport dstport protocol packets bytes start end action log_status
```

For example:

```
2 99999XXXXX eni-02f03c2880e4aaa3 10.0.1.70 10.0.1.11 9999 63030 6 6556 262256 1554562460 1554562475 ACCEPT OK
```

... and we want to use values of `packets` and `bytes` as metrics across these dimensions: `action`, `interface_id`, and `dstaddr`.

To reference the `packets` and `bytes` fields by name, as `'packets'` and `'bytes'`, our Pipeline will need a Parser Function before the Publish Metrics Function.

Parser Function

Filter: Set as needed Operation mode: Extract Type: Extended Log File Format (automatically set when specifying a library) Library: AWS VPC Flow Logs Source: `_raw` (No need to specify any other fields.)

Publish Metrics Function

Below, the `metric_name` prefix was arbitrarily chosen. Because there is no JavaScript expression to evaluate - this is literal text - the strings specified for the **Metric name expression** will be identical to those in the final metrics data sent to the Destination. See Raw Output below.

Metrics

Event Field Name	Metric Name Expression	Metric Type
`bytes`	`` `metric_name.bytes` ``	**Gauge**
`packets`	`` `metric_name.packets` ``	**Gauge**

Dimensions

```
action interface_id dstaddr
```

All specified dimension names must align with those from the original event. When you preview the Function's output, the metrics and dimensions will all have special highlighting to separate them from other fields. Additional highlighting is used to differentiate the metrics from the dimensions. (If one or more metrics/dimensions are not highlighted as expected, check the Function's configuration.)

Raw Output

```
metric_name.bytes:262256|g#action:REJECT,interface_id:eni-
02f03c2880e4aaa3,dstaddr:10.0.1.11

metric_name.packets:6556|g#action:REJECT,interface_id:eni-
02f03c2880e4aaa3,dstaddr:10.0.1.11
```



Compatible Destinations

All text after the `#` symbol represents the dimensions as key-value pairs. In order for dimension data to be included in metrics, the Destination type cannot be standard **StatsD**. However, **StatsD Extended**, **Splunk**, and **Graphite** do support dimensions.

Formatted Output


```
{
  "action": "REJECT",
  "interface_id": "eni-02f03c2880e4aaa3",
  "dstaddr": "10.0.1.11",
  "metric_name.bytes": 262256,
  "metric_name.packets": 6556,
}
```


Copy

Scenario B:

Assume that we want to extract some metrics from specific fields in PANOS logs, whose events have the following structure:

```
future_use_0,receive_time, serial_number, type, threat_content_type, future_use_1,
generated_time, source_ip, destination_ip, nat_source_ip, nat_destination_ip,
rule_name, source_user, destination_user, application, virtual_system, source_zone,
destination_zone, inbound_interface, outbound_interface, log_action, future_use_2,
session_id, repeat_count, source_port, destination_port, nat_source_port,
nat_destination_port, flags, protocol, action, bytes, bytes_sent, bytes_received,
```

packets, start_time, elapsed_time, category, future_use_3, sequence_number, action_flags, source_location, destination_location, future_use_4, packets_sent, packets_received, session_end_reason, device_group_hierarchy_level_1, device_group_hierarchy_level_2, device_group_hierarchy_level_3, device_group_hierarchy_level_4, virtual_system_name, device_name, action_source, source_vm_uuid, destination_vm_uuid, tunnel_id_imsi, monitor_tag_imei, parent_session_id, parent_start_time, tunnel_type, sctp_association_id, sctp_chunks, sctp_chunks_sent, sctp_chunks_received

For example:

```
Jan 10 10:19:15 DMZ-internal.nsa.gov 1,2019/01/10 10:19:15,001234567890002,TRAFFIC,drop,2304,2019/01/10 10:19:15,209.118.103.150,160.177.222.249,0.0.0.0,0.0.0.0,InternalServer,,,not-applicable,vsys1,inside,z1-FW-Transit,ethernet1/2,,All traffic,2019/01/10 10:19:15,0,1,63712,443,0,0,0x0,udp,deny,60,60,0,1,2019/01/10 10:19:15,0,any,0,0123456789,0x0,Netherlands,10.0.0.0-10.255.255.255,0,1,0,policy-deny,0,0,0,0,,DMZ-internal,from-policy,,,0,,0,,N/A,0,0,0,0,1202585d-b4d5-5b4c-aaa2-d80d77ba456e,0
```

Our goal is to use the four values of `bytes_sent`, `bytes_received`, `packets_sent`, and `packets_received` as metrics across these dimensions: `destination_ip`, `inbound_interface`, `outbound_interface`, and `destination_port`.

Here again, our Pipeline will need a Parser Function before the Publish Metrics Function.

Parser Function

Filter: Set as needed Operation mode: Extract Type: Extended Log File Format (automatically set when specifying a Library) Library: Palo Alto Traffic Source: `_raw` (No need to specify any other fields.)

Publish Metrics Function

Set up the Publish Metrics Function as follows.

Metrics

Event Field Name	Metric Name Expression	Metric Type
`bytes_sent`	`` `metric.${host}.bytes_sent` ``	Counter
`bytes_received`	`` `metric.${host}.bytes_rcvd` ``	Counter

Event Field Name	Metric Name Expression	Metric Type
packets_sent	`metric.${host}.pkts_sent`	Counter
packets_received	`metric.${host}.pkts_rcvd`	Counter

Added Dimensions

destination_ip, inbound_interface, outbound_interface, destination_port

Raw Output

```
metric.10.10.12.192.bytes_sent:60|c|#destination_ip:160.177.222.249,inbound_interface:et
metric.10.10.12.192.bytes_rcvd:0|c|#destination_ip:160.177.222.249,inbound_interface:eth
metric.10.10.12.192.pkts_sent:1|c|#destination_ip:160.177.222.249,inbound_interface:ethe
metric.10.10.12.192.pkts_rcvd:0|c|#destination_ip:160.177.222.249,inbound_interface:ethe
```

Here again, all text after the # symbol represents the dimensions as key-value pairs. (See the Compatible Destinations note above.) Unlike the first example, this example uses JavaScript expressions, which you can see evaluated in the raw output where the ${host} has been converted to 10.10.12.192.

15.28. Redis

The Redis Function interacts with Redis stores, setting and getting key-hash and key-value combinations. It uses in-memory caching of these key pairs to enable large lookup tables that would be cumbersome with a CSV or binary lookup file.

You can use Cribl Edge Collectors (for example, a REST Collector) to retrieve reference data from desired endpoints, and then use this Function to store the data on Redis and retrieve it to enrich your data. By default, Cribl Edge does not cache the data returned from this Redis Function, but you can enable client-side caching.

Your Redis Function can interact with three kinds of Redis instances: Standalone (the default); Redis Cluster, which scales horizontally; or Redis Sentinel, a distributed system that provides high availability.

Managing Redis Connections

Using Redis Functions can create a large number of TCP connections to the Redis server ("Redis connections" in short). This is because Cribl Edge creates a separate TCP connection to Redis for every Redis Function and every *reference to* a Redis Function. This means that certain usage patterns will increase the number of Redis Connections, including:

- Sending the output of a Chain Function to a regular Pipeline that includes a Redis Function.
- Using a Redis Function in a Source's pre-processing Pipeline.
- Using a Redis Function in a Destination's post-processing Pipeline.

To limit the number of Redis connections, you can *reuse* connections that share certain properties. These properties depend on the type of the Redis instance:

- Standalone: username, password, and URL.
- Cluster: username, password, and all root node hosts and ports.
- Redis Sentinel: username, password, master name, and all root node hosts and ports.

To enable reusing Redis connections:

1. In the sidebar, select **Fleets** and select the Fleet you want to configure.
2. On the **Fleet** submenu, select **Fleet Settings**.
3. Select **General Settings** > **Limits** > **Redis** > **Connections**.
4. Toggle **Reuse Redis connections** on.
5. Configure **Connection limit** and confirm with **Save**.

In a Single-instance deployment, the Redis connections settings are located in the main **Settings** menu.

When you enable **Reuse Redis connections**, Cribl Edge attempts to reuse an existing Redis connection instead of creating a new one if:

- A new Redis connection is necessary, and
- The new connection's properties match those of an existing, open connection.

The example below illustrates how powerfully this can affect the demands Cribl Edge places on a Redis server.

Determining Number of Connections

Connection limit is the upper limit of connections to Redis that Cribl Edge can establish before starting to reuse them. To determine the number to configure for your deployment, consider among others:

- Type of Redis instance you're connecting to (Standalone, Cluster, or Redis Sentinel)
- Size of the Redis server
- Number of connections the Redis server is configured to allow
- Number of other clients using the same Redis server

If you already have Redis connections established, be careful not to reduce **Connection limit** *below* the current connection number.

Example: Managing Redis Connections

Referencing a Redis Function from a Pipeline and/or another Function can dramatically multiply the number of Redis connections created, unless we mitigate this effect by applying the **Reuse Redis connections** control. Let's see how this plays out in a detailed example. Suppose we do the following:

1. Create a Pipeline (we'll call it "Pipeline Zero") that contains 10 Redis Functions that connect to the same Redis server, with identical connection properties. Cribl Edge will create 10 TCP connections to the Redis server ("Redis connections").
2. Reference Pipeline Zero in a Chain Function that's in another Pipeline. This creates 10 more connections, for a total of 20 Redis connections.
3. Reference Pipeline Zero in an additional, separate Chain Function that's in yet another Pipeline, creating 10 more connections, for a total of 30 Redis connections.
4. Use Pipeline Zero as a pre-processing Pipeline for a Source, creating 10 more connections, for a total of 40 Redis connections.

5. Use Pipeline Zero as a post-processing Pipeline for a Destination, creating 10 more connections. Now we have a grand total of 50 Redis connections.

Now consider the effect of these Pipelines within Edge Nodes and their Worker Processes. For a typical deployment scenario with 10 Edge Nodes, each of which have 10 Worker Processes, the calculation looks like this:

- 50 connections times 10 Worker Processes = 500 connections, times 10 Edge Nodes = 5000 connections.

A single Redis server can be hard-pressed to handle so many connections. Apart from that, the number of connections is unnecessarily high for the expected throughput. Let's use **Reuse Redis connections** to reduce the number of Redis connections.

Enable **Reuse Redis connections** and set **Connection limit** to `4`. Now revise the calculation:

- 4 connections times 10 Worker Processes = 40 connections, times 10 Edge Nodes = 400 connections.

This reduces the number of connections by better than a factor of 10, making our deployment much less likely to overwhelm the Redis server.

Usage

Filter: Filter expression (JavaScript) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Simple description of this Function. Defaults to empty.

Final: Toggle on to stop feeding data to the downstream Functions. Default is toggled off.

Result field: Name of the field in which to store the returned value. (Leave empty to discard the returned value.)

Command: Redis command to perform. Required. (A complete list of Redis commands is at: https://redis.io/commands.)

Key: A JavaScript expression to compute the value of the key to operate on. Can also be a constant such as `username`. This is a required field. Select the icon at right to open a validation modal.

Args: A JavaScript expression to compute arguments to the operation. Select the icon at right to open a validation modal. When the expression you enter is a plain value or an array, it resembles the options and arguments of a Redis command in the Redis CLI.

For example:

- Suppose you want to `SET` a key named `_mascot`. And, of the options that the `SET` command takes, you want to use two: `GET` and `EX`. The first, `GET`, takes no arguments; the second, `EX`, takes an integer in seconds.

- You want to set `_mascot` to the value `goat`, and `EX` to 5 seconds. Here is the array you'd enter in the Args column:
  `['goat', 'GET','EX',5 ]`

Deployment Type

From the drop-down, select `Standalone`, `Cluster`, or `Sentinel`. Some of these options display additional controls below. Hostnames must be JavaScript expressions (which can evaluate to a constant value), enclosed in quotes or backticks. They can be evaluated only at init time.

Standalone

This is the default deployment type. Enter a **Redis URL** to connect to. The format is:
`[redis[s]:]//[[user][:password@]][host][:port][/db-number][?db=db-number[&password=bar[&option=value]]]`

Example:
`redis://user:secret@localhost:6379/0?foo=bar&qux=baz`

Example with no `user` specified:
`redis://secret@localhost:6379/0?foo=bar&qux=baz`

Standalone with TLS

To enable TLS, specify a Redis URL that starts with `rediss://` rather than `redis://`. This URL reveals the **TLS** section of the config UI.

Example:
`rediss://user:secret@localhost:6379/0?foo=bar&qux=baz`

Example with no `user` specified:
`rediss://secret@localhost:6379/0?foo=bar&qux=baz`

Cluster

Specify **Root nodes** that the cluster will connect to. A single node is sufficient, but Cribl recommends listing two or more. Each of the **Root Nodes** consists of a **Hostname** and a **Port**.



In multi-key commands, ensure that all keys in the command are in the same hash slot. Otherwise, the command will fail. As a workaround, you can split the multi-key command into separate commands, where each command operates only on keys from a single hash slot.

Scale Reads determines which nodes in the Redis cluster should handle read commands. This setting helps optimize read performance and manage load distribution across the cluster. The available options are:

- **Masters**: Directs all read commands to master nodes only. This ensures strong consistency but may limit read scalability.
- **Replicas**: Directs all read commands to replica nodes only. This can improve read scalability but may result in stale data due to eventual consistency.
- **Masters and Replicas**: Distributes read commands across both master and replica nodes. This option balances load and improves read performance but may occasionally return stale data.

Sentinel

Specify a **Master group name**, plus every **Sentinel** in your Redis deployment. Each **Sentinel** consists of a **Hostname** and a **Port**.

Authentication Method

Use the **Authentication method** buttons to select one of the following options, some of which display additional controls below.

- **None**: Select this option where authentication either is not required, or is provided in the URL.
- **Basic**: This displays **Username** and **Password** fields for you to enter your Redis credentials.
- **User Secret**: This option exposes a drop-down in which you can select a stored text secret that references a Redis username and password, as described above. A **Create** link is available to store a new, reusable secret.
- **Admin Secret**: This option exposes a drop-down in which you can select a stored text secret that references a Redis admin password. A **Create** link is available to store a new, reusable secret.

TLS

Validate server certs: Reject certificates that are **not** authorized by a CA in the **CA certificate path**, nor by another trusted CA (such as the system's CA). Defaults to toggled on.

Server name (SNI): Server name for the SNI (Server Name Indication) TLS extension. This must be a host name, not an IP address.

Certificate: Select an existing certificate name from the drop-down or add a new certificate by selecting **Create**. See Add a TLS Certificate to a Fleet for details.

CA certificate path: Path on client containing CA certificates (in PEM format) to use to verify the server's cert. Path can reference `$ENV_VARS`.

Private key path (mutual auth): Path on client containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Certificate path (mutual auth): Path on client containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Passphrase: Passphrase to use to decrypt private key.

Minimum TLS version: Optionally, select the minimum TLS version to use when connecting. The minimum TLS version for Redis is version 6.

Maximum TLS version: Optionally, select the maximum TLS version to use when connecting.



Minimum TLS Requirement

The minimum Redis version required for TLS support is version 6.

Advanced Settings

Blocking time limit: Maximum amount of time (in seconds) before assuming that Redis is down and passing events through. Defaults to `60` seconds. Use `0` to disable timeouts.

Client-side cache: Toggle on to enable. Then navigate to **Settings** > **General Settings** > **Limits** > **Redis Cache** to configure the cache, as described in the next section.

Overall Redis Cache Settings

Together, these settings determine how the Cribl Edge client-side cache behaves. We'll start with definitions:

Key TTL in seconds: The maximum time-to-live of a key in the cache, in seconds. `0` means no limit. Defaults to 10 minutes.

Max # of keys: The maximum number of keys the cache can contain. `0` (the default) means no limit.

Max cache size (bytes): The maximum number of bytes the cache can contain. `0` (the default) means no limit.

Service period (seconds): The interval at which the cache checks whether any keys need to be evicted. Defaults to 30 seconds.

Server assisted: This feature requires Redis version 6 or newer. Toggle on to enable server-assisted caching and display the **Client tracking mechanism** drop-down. See the Tradeoffs section for an explanation of the available options.

Now here's the context:

For each Redis instance your Redis Functions interact with, Cribl Edge will operate one client-side cache. That cache will be shared between all the Redis Functions instances configured to interact with it. This is why the settings are in a central location, and why you cannot configure the cache at the individual function level.

Once per **Service period**, the cache checks whether any keys need to be evicted. The cache will evict:

- The least-recently-accessed key, if the cache has reached either (1) its **Max cache size (bytes)** or (2) its **Max # of keys**.
- Any key that exceeds its **Key TTL in seconds**.
- Any key that is stale according to a notification to Cribl Edge from your Redis instance. Redis sends these notifications (a.k.a. invalidations, or invalidation messages) only when you have enabled **Server assisted** caching.

The overall benefit that client-side caching provides is having fewer stale keys in your cache for less time. The greater the proportion of `GET`-type, as compared to `SET`-type (write) commands your Cribl Edge-Redis interactions have, the greater that benefit will be. When your Cribl Edge Pipelines either have no `SET`-type operations, or use filters to keep those operations to a minimum, you should consider client-side caching. In sum, you must decide whether more accurate data in your cache is worth the cost of memory and processing.

Tradeoffs Inherent in Server-Assisted Caching

Consider the common scenario where other applications besides Cribl Edge are writing to your Redis instance. Say one of those applications modifies a key's value. Now the value that Cribl Edge client-side cache has for that key is stale. Cribl Edge does not "know" about this, and retains the stale value until its TTL is up. This degrades the overall accuracy of data from the client-side cache.

Server-assisted caching is a remedy for this problem. With server-assisted caching enabled, your Redis instance will notify your client-side cache when Redis determines that a key's value has changed. Then the Cribl Edge client-side cache will immediately evict the stale key. Now the client-side cache is more accurate. Redis also has documentation about server-assisted client-side caching.

There are two mutually-exclusive ways you can have Redis notify Cribl Edge. These are the two modes (delivery mechanisms) available in the **Client tracking mechanism** drop-down:

1. In **Default** mode, the Redis server remembers which keys the Cribl Edge client has requested. Redis only notifies Cribl Edge when the value of one of **those** keys has changed. This is most efficient for Cribl Edge but needs more memory and some extra processing time from the Redis server.
2. In **Broadcast** mode, the Redis server need not remember which keys came from where, because it simply notifies Cribl Edge when the value of **any** key has changed. This is most efficient for Redis, but now the Cribl Edge client needs to inspect every key that Redis says has changed. This consumes some extra processing time on the client, and generates extra network activity to accommodate notifications about keys that Cribl Edge never sees.

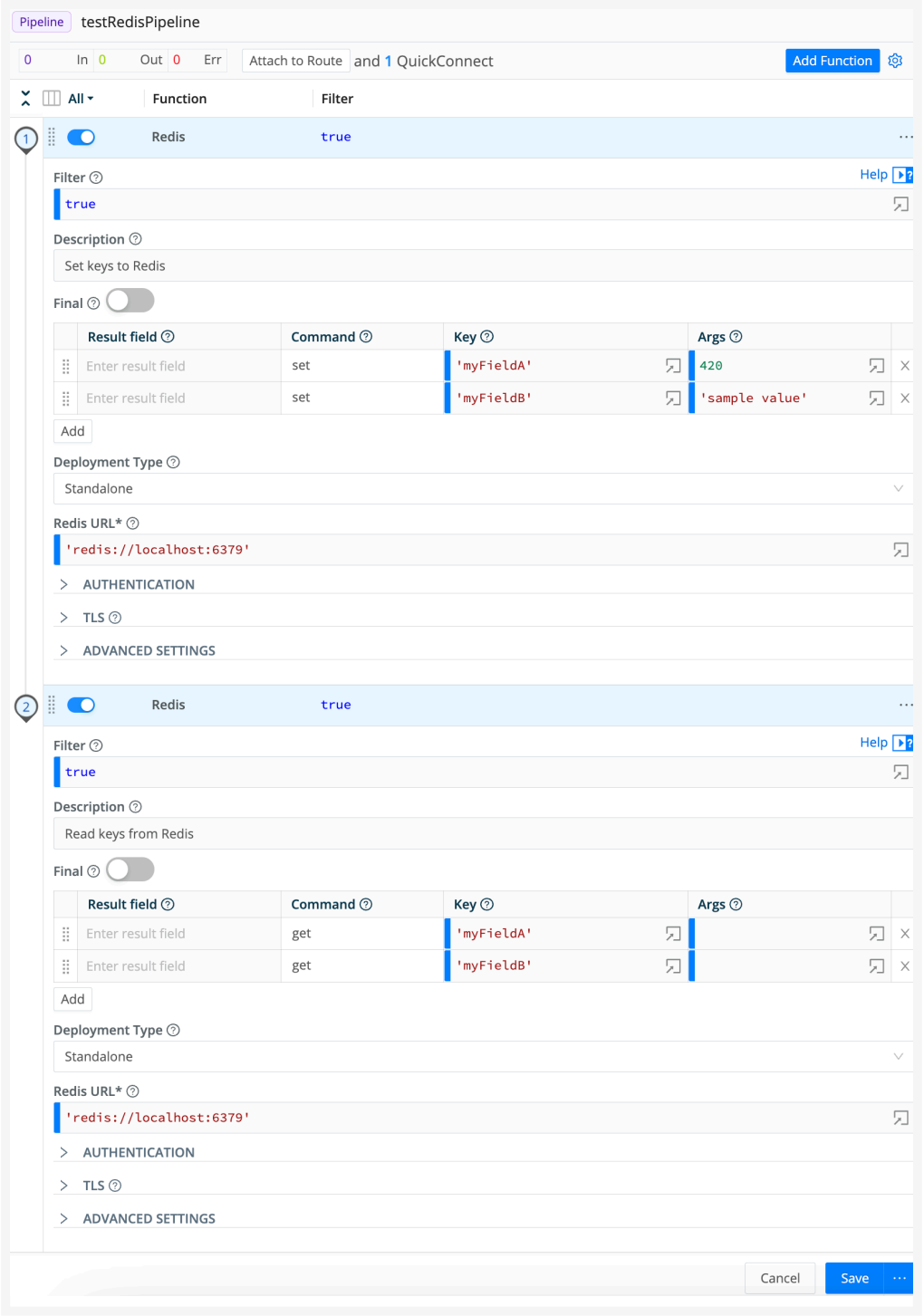
You should select whichever of these mechanisms best suits your environment. Does it make more sense to impose the heavier burden on the Redis server (as in **Default** mode) or on Cribl Edge (as in **Broadcast** mode)?

In both scenarios, for a given key, any write operation from any application (including Cribl Edge) that's interacting with Redis, causes Cribl to evict the key from the client-side cache when Redis notifies Cribl.

Examples

Scenario A: Set and Get

This Pipeline demonstrates the use of a pair of Redis Functions. The first Function sets two key-value pairs in Redis. The second Function gets their values, by key, into two corresponding new **Result field**s.



Redis set and get Functions

Redis Function #1

Description: `Set keys to Redis`

Command: `set` **Key:** `'myFieldA'` **Args:** `420`

Command: `set` **Key:** `'myFieldB'` **Args:** `'sample value'`

Redis Function #2

Description: `Read keys from Redis`

Result field: `myField_AA` **Command**: `get` **Key**: `'myFieldA'`

Result field: `myField_BB` **Command**: `get` **Key**: `'myFieldB'`

Scenario B: Multiple-Argument Arrays

This example demonstrates how to configure a Redis Function that supplies an array of multiple arguments to Redis commands (in this example, `lset` and `lrange`).



Redis Function, arrays of multiple arguments, and sample output

Redis Function

Description: `Push arrays of multiple arguments to Redis`

Result field: `rs1` **Command**: `rpush` **Key**: `"mylist"` **Args**: `'one'`

Result field: `rs2` **Command**: `rpush` **Key**: `"mylist"` **Args**: `'two'`

Result field: `rs3` **Command**: `rpush` **Key**: `"mylist"` **Args**: `'three`

Result field: `rs4` **Command**: `lset` **Key**: `"mylist"` **Args**: `[0,'four']`

Result field: `rs5` **Command**: `lset` **Key**: `"mylist"` **Args**: `[-2,'five']`

Result field: `rs6` **Command**: `lrange` **Key**: `"mylist"` **Args**: `[0,-1]`

Try This at Home

The Pipeline below contains only this example Function. You can import it into your own Cribl Edge environment, fill in the `url` with your own credentials, and further modify it to meet your needs.

redis-multiple-args.json

15.29. Regex Extract

The Regex Extract Function extracts fields using named capture groups. Fields that start with `__` (double underscore) are special in Cribl Edge. They are ephemeral: any Function downstream can use them, but they will be serialized only to Cribl internal Destinations.



This Function uses the JavaScript (ECMAScript) regex engine. If you test patterns in external tools such as Regex101, select the **ECMAScript (JavaScript)** flavor. For more information, see Regex Flavor.

Usage

Filter: Filter expression (JavaScript) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Simple description of the Function. Defaults to empty.

Final: Toggle on to stop feeding data to the downstream Functions. Default is toggled off.

Regex: Regex literal. Must contain named capturing groups: `(?<foo>bar)`. Can contain special `_NAME_N` and `_VALUE_N` capturing groups, which extract **both the name and value** of a field: `(?<_NAME_0>[^\s=]+)=(?<_VALUE_0>[^\s]+)`. Defaults to empty. You can type in a pattern, or use one of the predefined Regexes. See Examples below.

Additional regex: Select **Add Regex** to chain extra regex conditions.

Source field: Field on which to perform regex field extraction. This field supports nested addressing. Defaults to `_raw`.

Overwrite existing fields: Whether to overwrite existing event fields with extracted values. If toggled off (default), the function will convert existing fields to an array. If toggled on, Regex Extract will create array fields if applied multiple times, or if fields exist. For example, if you extract `src_ip` in an input Pipeline where the assigned value is `10.1.2.2`, and in a processing Pipeline with a value of `10.2.3.3`, then the resulting field is `["10.1.2.2", "10.2.3.3"]`.

Advanced Settings

Max exec: The maximum number of times to apply the **Regex** to the source field when the global flag is set, or when using `_NAME_N` and `_VALUE_N` capturing groups. Named capturing groups will always use a value of `1`. Defaults to `100`.

Field name format expression: JavaScript expression to format field names when using the `_NAME_n` and `_VALUE_n` capturing groups. For example, to append `XX` to all field names, use: `` `${name}_XX` `` (backticks are literal). If not specified, names will be sanitized using regex: `/^[_0-9]+|[^a-zA-Z0-9_]+/g`. The **original** field name is in the global `name`. You can access other fields' values via `__e.<fieldName>`.

Examples

To test Regex Extract Functions use Data Preview to see the events as they flow into and out of the Pipeline.

Example 1: Single Field from Simple Event

Assume a simple event that looks like this: `metric1=23 metric2=42 dc=23 abc=xyz`

Extract **only** the `metric1` field:

Regex: `metric1=(?<metric1>\d+)` **Result**: `metric1:"23"`

Example 2: Extracting Multiple Fields

Use this sample data:

```
462559d4a487[471]: 172.23.0.6 - - [26/Feb/2024:20:22:38 +0000] "GET /catalog/view/ja\
```

Copy

Use a regex to extract the fields you know are in the event.

Regex: `^(?<container_id>[^\s]+)\[(?<process_id>\d+)\]:\s+(?<remote_host>[^\s]+)\s+(?<remote_user>-)\s+(?<auth_user>-)\s+\[(?<timestamp>[^\]]+)\]\s+"(?<request_method>\w+)\s+(?<requested_url>[^\s]+)\s+(?<http_version>[^"]+)"\s+(?<status>\d+)\s+(?<bytes>.+)$`

Results:



Example 2 results

Example 3: Key-Value Pairs from Multiple Fields

Use this sample data:

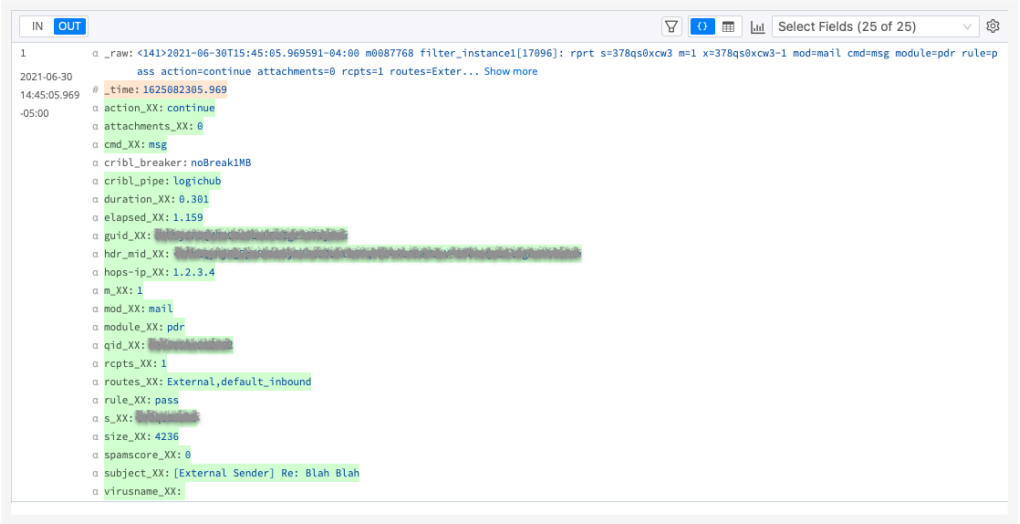
```
rec_type=71 rec_type_simple=RNA dest_port=443 snmp_out=0 netflow_src="00000000-0000-0
```

Copy

Use a regex to extract **all** k=v pairs, then use **Field Name Format Expression** to append an `_XX` suffix to each extracted field:

Regex: `(?<_NAME_0>[\w-]+)="?(?<_VALUE_0>(?<=")[^"]*|\S*)` **Field Name Format Expression**: `${name}_XX`

Results:



Example 3 results

Example 4: Multi-Stage Extraction, Complex Events

This example builds on the syntax in Example 3, to tackle a more complex event structure.

In the right **Sample Data** pane, select **Paste** and insert the following sample:

Sample Data

```
<134>1 2020-12-22T17:06:08Z CORP_INT_NLB CheckPoint 18160 - [action:"Accept"; conn_di
```

Copy

This event is from a CheckPoint Firewall CMA system. With this type of event structure, properly extracting each event field into a separate metadata field requires two-stage processing. So we'll use two Regex Extract Functions.

The first Regex Function splits the event to separate the actual data from the header information. We'll split after the `CheckPoint 18160` string, by capturing everything between the `[` and `]`:

Regex: `\[(?<__fields>.*)\]` **Source**: `_raw`

Next, add this second Regex Extract Function to extract all k=v pairs:

Regex: `(?<_NAME_0>[^ :]+):(?<_VALUE_0>[^;]+);` **Source**: `__fields`

Results:



Example 4 results

For further examples, see Using Cribl to Analyze DNS Logs in Real Time - Part 2.

15.30. Regex Filter

The Regex Filter Function filters out events based on regex matches.



This Function uses the JavaScript (ECMAScript) regex engine. If you test patterns in external tools such as Regex101, select the **ECMAScript (JavaScript)** flavor. For more information, see Regex Flavor.

Usage

Filter: Filter expression (JavaScript) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Simple description of this Function. Defaults to empty.

Final: Toggle on to stop feeding data to the downstream Functions. Default is toggled off.

Regex: Regex to test against. Defaults to empty. You can type in a pattern, or use one of the predefined Regexes.

Additional regex: Select **Add Regex** to chain extra regex conditions.

Field: Name of the field to test against the regex. Defaults to `_raw`. Supports nested addressing.

Examples

For a usage example, see our Regex Filtering topic.

15.31. Rename

The Rename Function changes field names or reformats their names (such as normalizing names to camelcase).

You can use Rename to change specified fields (much like the Eval Function), or for bulk renaming based on a JavaScript expression (much like the Parser Function). Compared to these alternatives, Rename offers a streamlined way to alter only field names, without other effects.

Usage

Filter: Filter expression (JavaScript) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Optionally, enter a simple description of this step in the Pipeline. Defaults to empty.

Final: Toggle on to stop feeding data to the downstream Functions. Default is toggled off.

Parent fields: Specify fields whose children will inherit the **Rename fields** and **Rename expression** operations. Supports wildcards. If empty, only top-level fields will be renamed.



This Function cannot operate on internal fields whose names begin with double underscores (`__`).

Note that you can still use `__e` to access fields within the `(context)` object, as long as those fields do not begin with double underscores. That's because `__e` is a special variable, not a field.

Rename fields: Each row here is a key-value pair that defines how to rename fields. The current name is the key, and the new name is the value. Select **Add Field** to add more rows.

- **Current Name**: Original name of the field to rename. You must quote literal identifiers (non-alphanumeric characters such as spaces or hyphens).
- **New Name**: New or reformatted name for the field. Here again, you must quote literals.

Rename expression: Optional JavaScript expression whose returned value will be used to rename fields. Use the `name` and `value` global variables to access fields' names/values. Example: `name.startsWith('data') ? name.toUpperCase() : name`. You can access the other fields' values via `__e.<fieldName>`.



A single Function can include both **Rename fields** (to rename specified field names) and **Rename expression** (to globally rename fields). However, the **Rename fields** strategy will execute first.

Advanced Settings

Parent field wildcard depth: For wildcards specified in **Parent fields**, sets the maximum depth within events to match and rename fields. Enter `0` to match only top-level fields. Defaults to `5` levels down.

Example

Change the `level` field, and all fields that start with `out`, to all-uppercase.

Example event:

```
{"inEvents": 622,
  "level": "info",
  "outEvents": 311,
  "outBytes": 144030,
  "activeCxn": 0,
  "openCxn": 0,
  "closeCxn": 0,
  "activeEP": 105,
  "blockedEP": 0
}
```

Copy

Rename Fields:

Current name: `level`
New name: `LEVEL` **Rename expression**: `name.startsWith('out') ? name.toUpperCase() : name`

Event after Rename:

```
{"inEvents": 622,
  "LEVEL": "info",
  "OUTEVENTS": 311,
  "OUTBYTES": 144030,
  "activeCxn": 0,
  "openCxn": 0,
  "closeCxn": 0,
  "activeEP": 105,
  "blockedEP": 0
}
```


Copy

Applications

1. Remove filename prefix `<myPrefix>`:

Rename expression: `name.replace(/<myPrefix>/, '')`

2. Add a wildcard to rename a set of fields named `json.record[0]`, `json.record[1]`, etc., preserving the variable numbers in the brackets:

Rename expression: `name.replace(/(json)\.(\w+)/,'MYNEWNAME-$2-$1')`

15.32. Rollup Metrics

The Rollup Metrics Function merges frequently generated incoming metrics into more manageable time windows.



Each Worker Process executes this Function independently on its share of events. For details, see Functions and Shared-Nothing Architecture.

Usage

Filter: Filter expression (JavaScript) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Optional description of this Function's purpose in this Pipeline. Defaults to empty.

Final: Toggle on to stop feeding data to the downstream Functions. Default is toggled off.

Dimensions: List of data dimensions across which to perform rollups. Supports wildcards. Defaults to `*` wildcard, meaning all original dimensions.

Time window: The time span over which to roll up (aggregate) metrics. Must be a valid time string (such as `10s`). Must match pattern: `\d+[sm]$`.



With high-cardinality data, beware of setting long time windows. Doing can cause high memory consumption and/or lost data, because memory is flushed upon restarts and redeployments.

Gauge update: The operation to use when rolling up gauge metrics. Defaults to **Last**; other options are **Maximum**, **Minimum**, or **Average**.

Examples

Scenario A:

Assume that you have metrics coming in at a rate that is too high. For example, Cribl Edge's internal metrics come in at a 2s interval.

To roll up these metrics to 1-minute granularity, you would set up the Rollup Metrics Function with a **Time Window** value of `60s`.

Scenario B:

Assume that you have metrics coming up with multiple dimensions - such as `host`, `source`, `data_center`, and `application`. You want to aggregate these metrics to eliminate some dimensions.

Here, you would configure Rollup Metrics Function with a **Time Window** value that matches the metrics' generation - such as `10s`. In the **Dimensions** field, you would remove the default `*` wildcard, and would specify only the dimensions you want to keep - such as `host`, `data_center`.

15.33. Sampling

The Sampling Function filters out events, based on an expression and a sampling rate.



Each Worker Process executes this Function independently on its share of events. For details, see Functions and Shared-Nothing Architecture.

Usage

Filter: Filter expression (JavaScript) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Simple description of this Function. Defaults to empty.

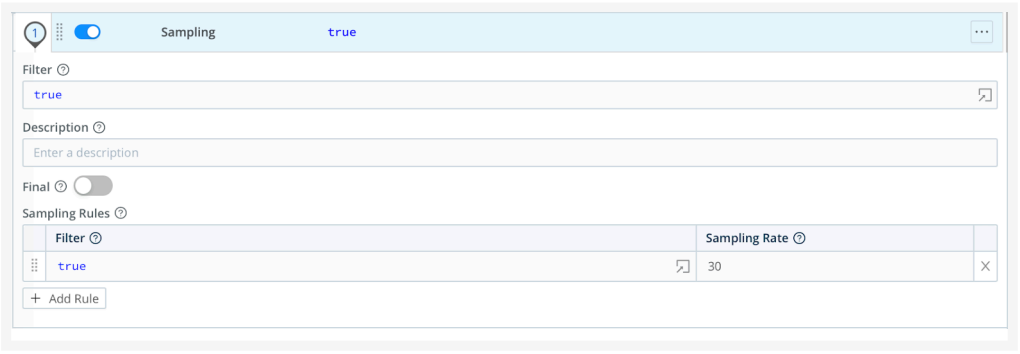
Final: Toggle on to stop feeding data to the downstream Functions. Default is toggled off.

Sampling rules: Events matching these rules will be sampled at the rates you specify:

- **Filter**: Filter expression matching events to be sampled. Use `true` to match all.
- **Sampling Rate**: Enter an integer `N`. (Defaults to `1`.) Sampling will pick `1/N` events matching this rule.

How It Works

Setting this Function's **Sampling rate** to `30` would mean that only 1 of every 30 events would be kept.



Sampling Rate

Let's assume that we save this setting, and then capture data from a datagen Source by selecting **Preview** > **Start a Capture** > **Capture**. In the **Capture Sample Data** modal, select: `100` seconds, `100` events, and **As they come in**. Then start the capture, and **Save as Sample File**.

Next, in the **Preview** pane, select **Simple** beside the new file's name. If you then select the **Basic Statistics** (chart) button, you should see that we've kept about 4 of the original 100 events, or close to 1 in 30.

	Full Event Length ⓘ	Number of Fields ⓘ	Number of Events ⓘ
IN	28.82KB	41	100
OUT	1.42KB	38	4
DIFF	↓ -95.08%	↓ -7.32%	↓ -96.00%

Examples and Scenarios

For usage examples, see these Better Practices topics:

- Sampling: Sampling verbose or voluminous data at ingest time, to make analysis and troubleshooting more efficient.
- Sample Logs: Sampling event data for efficient onboarding into Cribl Stream.
- Access Logs: Apache, ELB, CDN, S3, Etc.: Sampling voluminous logs from sources like Amazon S3, Amazon Cloudfront, and AWS ELBs.
- Firewall Logs: VPC Flow Logs, Cisco ASA, Etc.: Sampling voluminous logs from firewalls.

See Also

- Dynamic Sampling Function: Filter out events based on an expression, a sample mode, and the volume of events

15.34. SERIALIZE

Use the Serialize Function to convert an event's content into a predefined format.

Usage

Filter: Filter expression (JavaScript) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Simple description of this Function. Defaults to empty.

Final: Toggle on to stop feeding data to the downstream Functions. Default is toggled off.

Type: Data output format. Defaults to `CSV`. Options:

- **CSV**
- **Extended Log File Format** - Extended Log File Format.
- **Common Log Format** - Common Log Format.
- **Key=Value Pairs** - selecting this type displays additional options:
    - **Clean fields**: Toggle on to clean field names by replacing non-alphanumeric characters with `_`. This will also strip leading and trailing `"` symbols.
    - **Pair delimiter**: Delimiter used to separate `key=value` pairs. Defaults to a single space character. Multiple characters are allowed. Cannot be an equal sign (`=`).
- **JSON Object**
- **Delimited values** - selecting this type displays additional options:
    - **Delimiter**: Delimiter character to split value. If left blank, defaults to comma (`,`). You can also specify pipe (`|`) or tab characters.
    - **Quote char**: Character used to quote literal values. If left blank, defaults to `"`.
    - **Escape char**: Character used to escape delimiter or quote characters. If left blank, defaults to **Quote char**.
    - **Null value**: Field value representing the null value. These fields will be omitted. Defaults to: `-`

Library: Browse Parser/Formatter library.

Fields to serialize: List of fields to include in the output after this Function processes the event. For `CSV`, `ELFF`, `CLF`, and `Delimited values` formats, you must list each field explicitly. Wildcards (`*`) are not supported for these formats. For all other formats, you can use wildcard (`*`) field lists, which support both:

- Wildcards (`*`) to include multiple fields.
- Nested addressing, where you can reference a parent field and all its children using a wildcard. For example, if you convert `_raw` into an object, use `_raw*` to refer to `_raw` and all nested fields inside it. See Wildcard Lists for more information.



When using wildcard lists, all exclusions (such as `!fieldname`) must come before any inclusions (such as `fieldname` or `*`).

Source field: Field containing the object to serialize. Leave blank to serialize top-level event fields.

Destination field: Field to serialize the data into. Defaults to `_raw`.

Examples

Scenario A: JSON to CSV

Assume a simple event that looks like this: `{"time":"2019-08-25T14:19:10.240Z","channel":"input","level":"info","message":"initializing input","type":"kafka"}`

We want to serialize these fields: `_time`, `channel`, `level`, and `type` into a single string, in CSV format, stored in a new destination field called `test`.

To properly extract the key-value pairs from this event structure, we'll use a built-in Event Breaker:

1. Copy the above sample event to your clipboard.
2. In the **Preview** pane, select **Paste a Sample**, and paste in the sample event.
3. Under **Select Event Breaker**, choose **ndjson** (newline-delimited JSON), and select **Save as a Sample File**.

Now you're ready to configure the Serialize Function, using the settings below:

Type: CSV Fields to Serialize: `_time channel level type` Destination Field: `test` Source Field: [leave empty] **Result**: `test: 1566742750.24,input,info,kafka`

In the new `test` field, you now see the `time`, `channel`, `level`, and `type` keys extracted as top-level fields.

Scenario B: CSV to JSON

Let's assume that a merchant wants to extract a subset of each customer order, to aggregate anonymized order statistics across their customer base. The transaction data is originally in CSV format, but the statistical data must be in JSON.

Here's a CSV header (which we don't want to process), followed by a row that represents one order:

```
orderID,custName,street,city,state,zip 20200622102822,john smith,100 Main St.,Anytown,AK,99911
```

To convert to JSON, we'll need to first parse each field from the CSV to a manipulable field in the Pipeline, which the Serialize Function will be able to reference. In this example, the new manipulable field is `message`.

Use the `Parser` Function:

Filter: `true` Operation mode: `Extract` Type: CSV Source field: `_raw` Destination field: `message` List of fields: `orderID custName street city state zip`

Now use the Serialize Function:

Filter: `true` Type: `JSON` Fields to serialize: `city state` Source field: `message` Destination field: `orderStats`

15.35. SNMP Trap Serialize

Use the SNMP Trap Serialize Function to convert compliant events into SNMP traps for sending out through SNMP Trap Destinations that were modified within a Pipeline from an SNMP Trap Source. This ensures compatibility with existing SNMP infrastructure and supports integration with network management tools that use SNMP for monitoring and alerting.

Limitations

- The SNMP Trap Source the events came from must have **Include varbind types** enabled for this function to work. The SNMP Trap Serialize Function needs the varbind types to correctly serialize the events.
- For v2c events, `sysUpTime` and `snmpTrapOID` appear both at the top level and within varbinds. During serialization, the Function uses the values in varbinds as they reflect the event's generation time. If the values differ, varbinds take precedence. Modifying the top-level values will not affect serialization.

Usage

Filter: Filter expression (JavaScript) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Simple description of this Function. Defaults to empty.

Final: Toggle on to stop feeding data to the downstream Functions. Default is toggled off.

Enforce required fields: When toggled on (default), v2c and v3 traps must meet the following requirements:

- At least two varbinds must be present.
- The first varbind must have the `sysUpTime` OID (`1.3.6.1.2.1.1.3.0`) and type `67` (`TimeTicks`).
- The second varbind must have the `snmpTrapOid` OID (`1.3.6.1.6.3.1.1.4.1.0`) and type `6` (`OID`).

These requirements are mandated by RFC 3416 Section 4.2.6, RFC 1066 Section 5.1, and RFC 1905 Section 3.1.2. While traps can be serialized without these fields, downstream receivers may drop them. With this option enabled, non-compliant traps are dropped before being sent downstream.

Drop failed events: When toggled on (default), events that fail to serialize into valid traps are dropped entirely, preventing them from passing through the Pipeline. This is useful when sending to an SNMP Trap Destination, as invalid traps will not be sent.

When toggled off, the event will pass through without `__snmpRaw` and will include a `snmpSerializeErrors` field for troubleshooting.

Regardless of whether **Drop failed events** is toggled on or off, if an event fails to serialize, the `__snmpRaw` field is deleted, preventing the event from leaving the SNMP Trap Destination.

Recommended setup flow:

1. Get a sample of the traps to modify and configure the SNMP Trap Serialize Function.
2. Enable **Drop failed events** and run a Data Preview.
3. Modify the SNMP Trap Serialize Function and preceding Eval Function until no `snmpSerializeErrors` remain.
4. Disable **Drop failed events** when confident in the Pipeline setup and ready to flow actual data.

SNMPV3 User: SNMPv3 provides enhanced security by enabling user authentication and data encryption. These settings are crucial for ensuring that only authorized users can send SNMP traps and that the data remains secure during transmission.

- **Username**: The username for SNMP v3 authentication. Ensure the username matches the one configured on the SNMP manager.
- **Authentication protocol**: The protocol to use for SNMP v3 authentication. Options include `None`, `MD5`, `SHA1`, `SHA224`, `SHA256`, `SHA384`, and `SHA512`.
    - **V3 authentication key**: The key used for SNMPv3 authentication. This field appears when an **Authentication protocol** is selected.
    - **Privacy protocol**: The protocol to use for SNMPv3 encryption. This field appears when an **Authentication protocol** is selected, as it is not possible to have an encrypted trap without authentication. Options include:
        - `None`
        - `AES128`
        - `AES256b (Blumenthal)`
        - `AES256r (Reeder)`
    - **V3 privacy key**: The key used for SNMPv3 encryption. This field appears when a **Privacy protocol** is selected.

Maintain Original Source IP in SNMP Traps

This setup maintains the original source IP in SNMP traps by adding it to the varbinds array, ensuring accurate data transmission even when traps traverse network devices that alter IP headers.

Before the SNMP Trap Function in your Pipeline, use the Eval Function to append a new object to the varbinds array. The object includes an OID, type, and the value from the host field, effectively capturing the original source IP. For example:

- **Name**: `varbinds`
- **Value expression**: `[...varbinds, {oid: '.1.3.6.1.6.3.18.1.3', type: 64, value: __e.host}]`

Next, for the SNMP Trap Serialize Function we recommend enabling both **Enforce required fields** and **Drop failed events**. This will ensure that the SNMP trap is serialized correctly, enforcing necessary fields and dropping events that fail serialization.

Change Authorization and/or Privacy Level

To change the authorization or privacy level of an SNMP v3 trap, you must modify two settings:

- **SNMPV3 User**: Update the user configuration within the SNMP Trap Serialize Function.
- **msgFlags**: Set the appropriate value at the top level of the event in an Eval Function before the serialization.

The `msgFlags` value should be a string representation of the desired number:

- `00`: No authorization, no privacy
- `01`: Authorization, no privacy
- `03`: Authorization and privacy

Change the SNMPv3 User

To change the SNMPv3 user, you need to update the `msgUserName` field within the SNMP Trap object.

Before the SNMP Trap Function in your Pipeline, use the Eval Function to update `msgSecurityParams.msgUserName` to the new intended username. For example, **Eval expression**: `msgSecurityParams.msgUserName = 'newUser';`

15.36. Suppress

The Suppress Function either drops or tags specified events over a specified time period, based on evaluation of a key expression.



Each Worker Process executes this Function independently on its share of events. For details, see Functions and Shared-Nothing Architecture.

Usage

Filter: Filter expression (JavaScript) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Simple description of this Function. Defaults to empty.

Final: Toggle on to stop feeding data to the downstream Functions. Default is toggled off.

Key expression: Suppression key expression used to uniquely identify events to suppress. For example, `` `${ip}:${port}` `` will use the fields `ip` and `port` from each event to generate the key.

Number to allow: The number of events to allow per time period. Defaults to `1`.

Suppression period (sec): The number of seconds to suppress events after **Number to allow** events are received. Defaults to `30`.

Drop suppressed events: Specifies if suppressed events should be dropped, or just tagged with `suppress=1`. Defaults to toggled on, meaning drop.

Advanced Settings

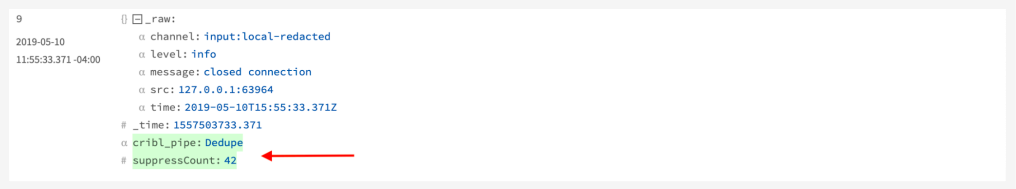
Cache size limit: The maximum number of keys that can be cached before idle entries are removed. Before changing the default `50000`, contact Cribl Support to understand the implications.

Suppression period timeout: The number of suppression periods of inactivity before a cache entry is considered idle. This defines a multiple of the **Suppression period (sec)** value. Before changing the default `2`, contact Cribl Support to understand the implications.

Num events to trigger cache clean-up: Check cache for idle sessions every N events when cache size exceeds the **Cache size limit**. Before changing the default `10000`, contact Cribl Support to understand the implications.

Seeing the Results

If you've enabled **Drop suppressed events**, such events will be omitted from logs as they exit this Function. However, the next event allowed through will include a `suppressCount: N` field, whose `N` value indicates the number of events dropped in the preceding **Suppression period**.



suppressCount shows number of events dropped

Examples

In the examples below, **Filter** is the Function-level Filter expression:

1. Suppress by the value of the `host` field: Filter: `true` Key expression: `host` Number to allow: `1` Suppression period (sec): `30`
   Using a datagen sample as a source, generate at least 100 events over 2 minutes.

Result: One event per unique `host` value will be allowed in every 30s. Events without a `host` field will **not** be suppressed.

2. Suppress by the value of the `host` and `port` tuple : Filter: `true` Key expression: `` `${host}:${port}` `` Number to allow: `1` Suppression period (sec): `300`

Result: One event per unique `host:port` tuple value will be allowed in every 300s.



Suppression will **also** apply to events without a `host` or a `port` field. The reason is that if `field` is not present, `` `${field}` `` results in the literal `undefined`.

3. To **guarantee** that suppression applies **only** to events with `host` and `port`, check for their presence using a Filter: Filter: `host && port!=null` Key expression: `` `${host}:${port}` `` Number to allow: `1` Suppression period (sec): `300`
4. Decorate events that qualify for suppression: Filter: `true` Key expression: `` `${host}:${port}` `` Number to allow: `1` Suppression period (sec): `300` Drop suppressed events: Toggle off.

Result: No events will be suppressed. But all qualifying events will gain an added field `suppress=1`, which can be used downstream to further transform these events.

For further use cases, see Cribl's Streaming Data Deduplication with Cribl blog post. For further details on filter usage, see Build Custom Logic to Route and Process Your Data: Filter Expressions.

15.37. Tee

The Tee Function tees events out to a command of choice, via `stdin`. The output is one JSON-formatted event per line. You can send the events to (for example) a local file on the Cribl Edge Worker. This can be useful in verifying the data being processed in a Pipeline.

The Filesystem/NFS Destination offers similar capability, but only after the data leaves the Pipeline. Tee, by comparison, can be inserted at any point in the Pipeline.

In Cribl.Cloud, the Tee Function is only available on customer-managed hybrid Edge Nodes.

Usage

Filter: Filter expression (JavaScript) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Simple description of this Function. Defaults to empty.

Final: Toggle on to stop feeding data to the downstream Functions. Default is toggled off.

Command: Command to execute and receive events (via `stdin`) - one JSON-formatted event per line.

Args: Select **Add Arg** to supply arguments to the command.

Restart on exit: Restart the process if it exits and/or we fail to write to it. Defaults to toggled on.

Environment variables: Environment variables to set or overwrite. Select **Add Variable** to add key-value pairs.

Communication Protocol

Data is passed to the command through its `stdin`, using the following protocol:

- First line: Metadata serialized in JSON, containing the following fields:
  - **format**: Serialization format for event. Defaults to `JSON`.
  - **conf**: Full Function configuration.
- Remaining: Payload.

Examples

Assume that we are parsing PANOS Traffic logs, and want to see how they look at a particular step in the processing Pipeline. We'll assume that the `Parser` Function is already in place, so we'll insert the Tee Function at any (arbitrary) later point in the Pipeline.

Scenario A:

The Tee Function itself requires only that we define the **Command** field. In this particular example, that **Command** will be `tee` itself.

We've also selected **Add Arg**, to specify a local output file in the resulting **Args** field. (A file path would normally be the first argument to a `tee` command executed from the command line. The Cribl Edge user must have write permission on the specified file path.)

Command: `tee`

Args: `/opt/cribl/foo.log`

In this first scenario, assume that we have the `Parser` configured to parse, but not keep any fields. After changes are deployed and PANOS logs are received, if we tail `foo.log`, we'd see the following:

```
Line 1: {"format":"json","conf":{"restartOnExit":true,"env":{},"command":"tee","args":["/opt/cribl/foo.log"]}
```

```
Line 2: {"_raw":"Oct 09 10:19:15 DMZ-internal.nsa.gov 1,2019/10/09 10:19:15,001234567890002,TRAFFIC,drop,2304,2019/10/09 10:19:15,209.118.103.150,160.177.222.249,0.0.0.0,0.0.0.0,InternalServer,,,not-applicable,vsys1,inside,z1-FW-Transit,ethernet1/2,,All traffic,2019/10/09 10:19:15,0,1,63712,443,0,0,0x0,udp,deny,60,60,0,1,2019/10/09 10:19:15,0,any,0,0123456789,0x0,Netherlands,10.0.0.0-10.255.255.255,0,1,0,policy-deny,0,0,0,0,,DMZ-internal,from-policy,,,0,,0,,N/A,0,0,0,0,1202585d-b4d5-5b4c-aaa2-d80d77ba456e,0","_time":1593185574.663,"host":"127.0.0.1"}
```

In Line 2 above, note that the `_raw` field makes up most of the contents, with only the `_time` and `host` fields added.

Scenario B:

Assume that we use the Tee Function, using the same **Command** and arguments, but we've modified the `Parser` Function to retain five fields: `receive_time`, `source_port`, `destination_port bytes_received`, and `packets_received`.

This time, if we tail `foo.log`, we'll see something like the following. If you compare this output to the previous output example, you'll notice the five fields appended to this event:

```
Line 3: {"_raw":"Oct 09 10:19:15 DMZ-internal.nsa.gov 1,2019/10/09
10:19:15,001234567890002,TRAFFIC,drop,2304,2019/10/09
10:19:15,209.118.103.150,160.177.222.249,0.0.0.0,0.0.0.0,InternalServer,,,not-
applicable,vsys1,inside,z1-FW-Transit,ethernet1/2,,All traffic,2019/10/09
10:19:15,0,1,63712,443,0,0,0x0,udp,deny,60,60,0,1,2019/10/09
10:19:15,0,any,0,0123456789,0x0,Netherlands,10.0.0.0-10.255.255.255,0,1,0,policy-
deny,0,0,0,0,,DMZ-internal,from-policy,,,0,,0,,N/A,0,0,0,0,1202585d-b4d5-5b4c-aaa2-
d80d77ba456e,0","_time":1593185606.965,"host":"127.0.0.1","receive_time":"2019/10/09
10:19:15","source_port":"63712","destination_port":"443","bytes_received":"0","packets_r
```



In this Function's **Command** field, you can specify commands other than `tee` itself. For example: By using `nc` as the command, and specifying `localhost` and a port number (as two separate arguments), you'll see event data being received via `nc` on the specified port.

15.38. UNROLL

The Unroll Function accepts an array field - or an expression to evaluate an array field - and breaks (or *unrolls*) the array into individual events.

Usage

Filter: Filter expression (JavaScript) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Simple description of this Function. Defaults to empty.

Final: Toggle on to stop feeding data to the downstream Functions. Default is toggled off.

Source field expression: Field in which to find/calculate the array to unroll. Example: `_raw`, `_raw.split(/\n/)`. Defaults to `_raw`.

Destination field: Field (within the destination event) in which to place the unrolled value. Defaults to `_raw`.

Example

Assume we want to break/unroll each line of this event:

Sample Event

```
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.5  38000  5356 ?        Ss    2018   2:02 /lib/systemd/systemc
root         2  0.0  0.0      0     0 ?        S     2018   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S     2018   1:51 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S<    2018   0:00 [kworker/0:0H]
root         7  0.0  0.0      0     0 ?        S     2018   3:55 [rcu_sched]
root         8  0.0  0.0      0     0 ?        S     2018   0:00 [rcu_bh]
```

Copy

Settings

Source field expression: `_raw.split(/\n/)`

The `split()` JavaScript method breaks `_raw` into an ordered set of substrings/values, puts these values into an array, and returns the array.

Destination field: `_raw`

Resulting Events

```
Event 1:
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND

Event 2:
root         1  0.0  0.5  38000  5356 ?        Ss    2018   2:02 /lib/systemd/systemc

Event 3:
root         2  0.0  0.0      0     0 ?        S     2018   0:00 [kthreadd]

Event 4:
root         3  0.0  0.0      0     0 ?        S     2018   1:51 [ksoftirqd/0]

Event 5:
root         5  0.0  0.0      0     0 ?        S<    2018   0:00 [kworker/0:0H]

Event 6:
root         7  0.0  0.0      0     0 ?        S     2018   3:55 [rcu_sched]

Event 7:
root         8  0.0  0.0      0     0 ?        S     2018   0:00 [rcu_bh]
```

Copy

15.39. XML Unroll

The XML Unroll Function accepts a proper XML event with a set of elements, and converts the elements into individual events.

Usage

Filter: Filter expression (JavaScript) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Simple description of this Function. Defaults to empty.

Final: Toggle on to stop feeding data to the downstream Functions. Default is toggled off.

Unroll elements regex: Path to the array to unroll. Example: `^root\.child\.ElementToUnroll$`

Copy elements regex: Regex matching elements to copy into each unrolled event. Example: `^root\.(childA|childB|childC)$`

Unroll index field: Cribl Edge will add a field with this name, containing the 0-based index at which the element was located within the event. In Splunk, this will be an index-time field. Supports nested addressing. Name defaults to `unroll_idx`.

Pretty print: Whether to pretty print the output XML.

Examples

Assume that the following sample is ingested as a single event:

sample.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<Parent>
    <myID>123456</myID>
    <branchLocation>US</branchLocation>
    <Child>
        <state>NY</state>
        <city>New York</city>
    </Child>
    <Child>
        <state>NJ</state>
        <city>Edgewater</city>
    </Child>
    <Child>
        <state>CA</state>
        <city>Oakland</city>
    </Child>
    <Child>
        <state>CA</state>
        <city>San Francisco</city>
    </Child>
</Parent>
```

Copy

If you insert this sample using **Preview** > **Add a Sample** > **Paste a Sample**, adjust Event Breaker settings to add the sample as a single event. One way to do this is to add a regex Event Breaker that (by design) will not match anything present in the sample. For example: `/[\n\r]+donotbreak(?!\s)/`. In current Cribl Edge versions, you can also use the built-in **Do Not Break** Ruleset.

Set up the XML Unroll Function using these settings:

Unroll elements regex: `^Parent\.Child$` **Copy elements regex**: `^Parent\.(myID|branchLocation)$`

Output 4 Events:

Resulting Events

```
# Event 1
<?xml version="1.0"?>
<Child>
  <myID>123456</myID>
  <branchLocation>US</branchLocation>
  <state>NY</state>
  <city>New York</city>
</Child>

# Event 2
<?xml version="1.0"?>
<Child>
  <myID>123456</myID>
  <branchLocation>US</branchLocation>
  <state>NJ</state>
  <city>Edgewater</city>
</Child>

# Event 3
<?xml version="1.0"?>
<Child>
  <myID>123456</myID>
  <branchLocation>US</branchLocation>
  <state>CA</state>
  <city>Oakland</city>
</Child>

# Event 4
<?xml version="1.0"?>
<Child>
  <myID>123456</myID>
```

15.40. Windows SID Lookup

The Windows SID Lookup Function translates Windows SID (security identifiers) into canonical account names for use in downstream Pipeline steps.

The Function uses the local Windows host where the Edge Node is running to resolve the SID, relying on OS resolution for domain objects.



The Windows SID Lookup Function works only in Cribl Edge on Windows.

Because Cribl Leader Nodes always run on Linux, previewing the results of the Windows SID Lookup Function from the Leader will not work. Instead, to preview the results, teleport into an Edge Node running on Windows.

How the Windows SID Lookup Function Works

The Windows SID Lookup Function returns:

- The corresponding domain or account value.
- An error message when a SID cannot be resolved.
- "Unsupported Platform" when used on a platform other than Windows.

Configure the Windows SID Lookup Function

To configure this Function:

1. Add a Windows Event Logs source to begin generating data. See Windows Event Logs for more information.
2. Add a new Pipeline or open an existing Pipeline. Add or capture sample data test your Pipeline as you build it. See Pipelines for more information.
3. At the top of the Pipeline, select **Add Function** and search for `Windows SID Lookup`, then select it.
4. In the **Windows SID Lookup** section, configure the following general settings:
    - **Filter:** A JavaScript filter expression that selects which data to process through the Function. Defaults to `true`, meaning it evaluates all data events. If you don't want to use the default, you can use a JavaScript expression to apply this Function to specific data events. See Build Custom Logic to Route and Process Your Data for examples.

- **Description:** A brief description of how this Function modifies your data to help other users understand its purpose later. Defaults to empty.
- **Final:** When enabled, stops data from continuing to downstream Functions for additional processing. Default: Off. Toggle **Final** on if Fold Keys is the last Function in your Pipeline.
- **Lookup fields**: Table of fields to place translated SID is. Select **Add Field** to add each field in a row with the following controls:
  - **Name**: Enter the name of the field to contain the translated SIDs.
  - **Value Expression**: Enter a JavaScript expression to extract the SID value(s) from the event field (typically `_raw`). Expressions can contain nested addressing. When you insert JavaScript template literals, strings intended to be used as values must be delimited with single quotes, double quotes, or backticks.
  - **Enabled**: Toggle this off to disable evaluating individual expressions, while retaining their configuration in the table. Useful for iterative development and debugging.

5. Test that you configured the Function correctly by comparing sample incoming data with outgoing data in the Data Preview pane and Pipeline Diagnostics tool. See Data Preview for more information.

15.41. Prometheus Publisher (Deprecated)



This Function was deprecated as of Cribl Edge 3.0, and has been removed from Cribl Edge as of v.4.0. Please instead use the Prometheus Destination to send metrics to Prometheus-compatible endpoints.

The Prometheus Publisher Function allows for metrics to be published to a Prometheus-compatible metrics endpoint. These can be upstream metrics received by Cribl Edge, or metrics derived from the output of Cribl Edge's `Publish Metrics` or `Aggregation` Functions. A Prometheus instance is responsible for collecting the metrics at that endpoint, and for performing its own processing of the metric data.

In the current Cribl Edge version, the endpoint is: `http://<worker_node_IP>:<api-port>/metrics`. Within Cribl Edge, that endpoint redirects from `http://<worker_node_IP>:9000/metrics` to `http://<worker_node_IP>:9000/api/v1/metrics`.



If used, this Function **must** follow any Publish Metrics or Aggregations Functions within the same Pipeline. This is to ensure that any data **not** originating from a metrics input is transformed into metrics format.

Usage

Filter: Filter expression (JavaScript) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Simple description of this Function. Defaults to empty.

Final: Toggle on to stop feeding data to the downstream Functions. Default is toggled off.

Fields to publish: Wildcard list of fields to publish to the Prometheus endpoint.

Advanced Settings

Batch write interval: How often, in milliseconds, the contents should be published. Defaults to `5000`.

Passthrough mode: Toggle off (default) to override the **Final** setting and suppress output to downstream Functions' Destinations. Toggle on to allow events to flow to consumers beyond the Prometheus endpoint. In effect, when previewing the pipeline output what you'll see is your event fields will have strikethrough font applied to them. This does not mean the Prometheus function is not matching your events but rather indicative of the Passthrough being disabled.

Update mode: Toggle off (default) to supporess output to downstream Functions' Destinations. (This overrides the **Final** setting.) Toggle on to allow events to flow to consumers beyond the Prometheus endpoint.

Example

This example uses the same PANOS sample data as the Publish Metrics Function, and is similarly preceded in a Pipeline by a Parser Function that extracts fields from the PANOS log.

Filter: Set as appropriate. **Fields to publish**: Set as appropriate. We'll use the default of `*` for this example. **Advanced settings**: Accept defaults.

After committing and deploying changes, you should be able to use a `curl` command (-L needed to follow the redirect mentioned above) to verify that metrics are being published, just a few seconds after data is ingested on an idle system.

curl output

```
$ curl -L http://<worker_node_IP>:9000/metrics
# TYPE perf_192_168_1_248_bytes_sent counter
metric_192_168_1_248_bytes_sent {destination_ip="160.177.222.249",inbound_interface='

# TYPE perf_192_168_1_248_bytes_rcvd counter
metric_192_168_1_248_bytes_rcvd {destination_ip="160.177.222.249",inbound_interface='

# TYPE perf_192_168_1_248_pkts_sent counter
metric_192_168_1_248_pkts_sent {destination_ip="160.177.222.249",inbound_interface="e

# TYPE perf_192_168_1_248_pkts_rcvd counter
metric_192_168_1_248_pkts_rcvd {destination_ip="160.177.222.249",inbound_interface="e
```

Copy

Now, we need to have Prometheus scrape the metrics. In this very basic example, you can add the target endpoint to the `prometheus.yml` file, under the `scrape_configs` -> `static_configs` section. Specify the endpoint in `IP:port` syntax, because Prometheus assumes (and requires) `/metrics` for all endpoints.

Restart Prometheus. Within just a few seconds, you should be able to use its query interface to retrieve metrics published by Cribl Edge.

15.42. Reverse DNS (Deprecated)

The Reverse DNS Function resolves hostnames from a numeric IP address, using a reverse DNS lookup.



This Function was deprecated as of v.2.4, and has been removed from Cribl Edge as of v.4.0. Please instead use the DNS Lookup Function's reverse lookup feature.

Usage

Filter: Filter expression (JavaScript) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Simple description of this Function. Defaults to empty.

Final: Toggle on to stop feeding data to the downstream Functions. Default is toggled off.

Lookup Fields

Lookup field name: Name of the field containing the IP address to look up.



If the field value is not in IPv4 or IPv6 format, the lookup is skipped.

Output field name: Name of the field in which to add the resolved hostname. Leave blank to overwrite the lookup field.

Reload period (minutes): How often to refresh the DNS cache. Use `0` to disable refreshes. Defaults to `60` minutes.

Example

Lookup field name: `dest_ip` **Output field name**: `dest_host` **Result**: See the `dest_ip` field, and the newly created `dest_host` field, in the events.


1
2020-06-29 12:51:03.551 -07:00
_raw: rec_type=71 rec_type_simple=RNA dest_port=443 snmp_out=0 netflow_src=00000000-0000-0000-0000-000000000000 ssl_server_cert_status="Not Checked" dest_ip=8.8.8.8 sec_intel_event=No mac_address=00:00:0... Show more
_time: 1593460263.551
app_proto: HTTPS
client_app: SSL client
client_version:
connection_id: 21378
cribl_breaker: Break on newlines
cribl_pipe: rev_dns
dest_autonomous_system: 0
dest_bytes: 3746
dest_host: dns.google

15.43. Trim Timestamp (Deprecated)

The Trim Timestamp Function removes timestamp patterns from events, and (optionally) stores them in a specified field.



This Function is deprecated as of Cribl Edge 4.7, and will be removed in the next major version. Please use the Eval Function instead.

This Function looks for a timestamp pattern that exists between the characters indicated by numeric `timestartpos` and `timeendpos` fields. **From events whose `timestartpos` value is set to `0`**, the Function removes `timestartpos` and `timeendpos` along with the timestamp pattern.

Usage

Filter: Filter expression (JavaScript) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Simple description about this step in the Pipeline. Defaults to empty.

Final: Toggle on to stop feeding data to the downstream Functions. Default is toggled off.

Field name: Name of field in which to save the timestamp. (If empty, timestamp will not be saved to a field.)

Example

Remove the timestamp pattern (indicated by `timestartpos` and `timeendpos`) from `_raw`, and stash it in a field called `time_field`.

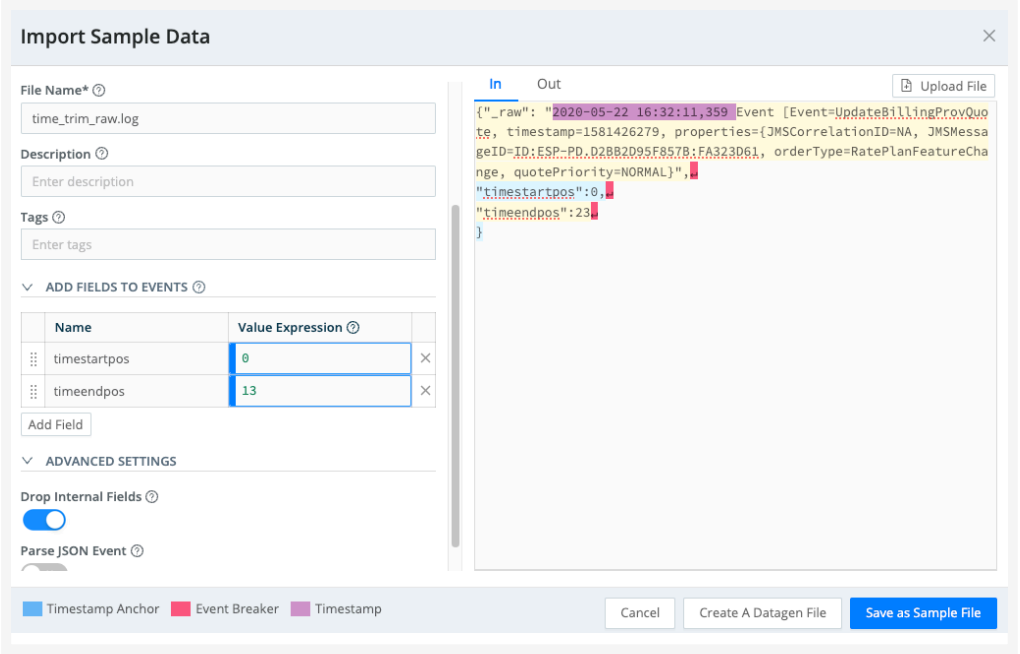
Field name: `time_field`

Example event before:

```
{"_raw": "2020-05-22 16:32:11,359 Event [Event=UpdateBillingProvQuote, timestamp=1581
"timestartpos":0,
"timeendpos":23
}
```

Copy

To create this example payload, we selected **Sample Data** > **Import**, pasted the `_raw` field's original contents into the resulting modal, and then added the two required position fields:



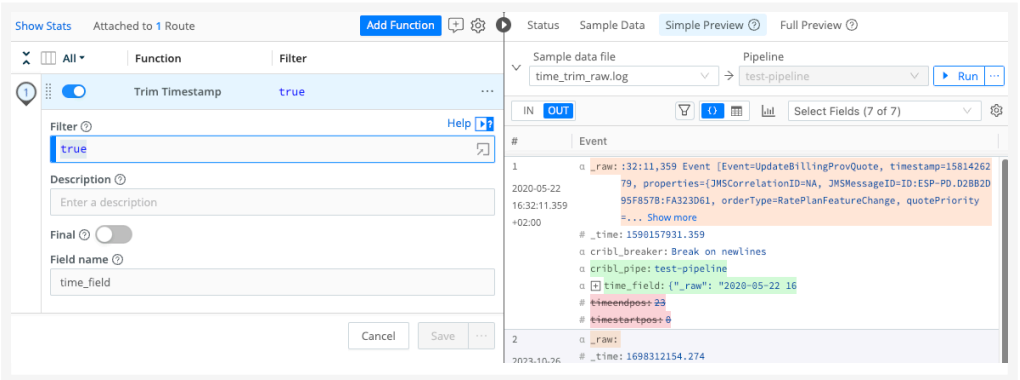
Example event setup

Example event after:

```
{"_raw": "Event [Event=UpdateBillingProvQuote, timestamp=1581426279, properties={JMSC
"time_field":"2020-05-22 16:32:11,359"
}
```

Copy

In the Preview pane's **OUT** view, the original timestamp has been removed from `_raw`, and lifted into the new `time_field` we specified in the Function. The `timestartpos` and `timeendpos` fields have been removed.



Example event, saved and transformed

16. Packs

Packs are self-contained bundles of Cribl Edge configurations. A Pack can include Routes, Pipelines, Sources, Destinations, and knowledge objects, grouped as a single portable unit that you can install, inspect, and reuse across Worker Groups and environments. You can create your own Packs or download pre-built configurations directly from the Cribl Packs Dispensary.

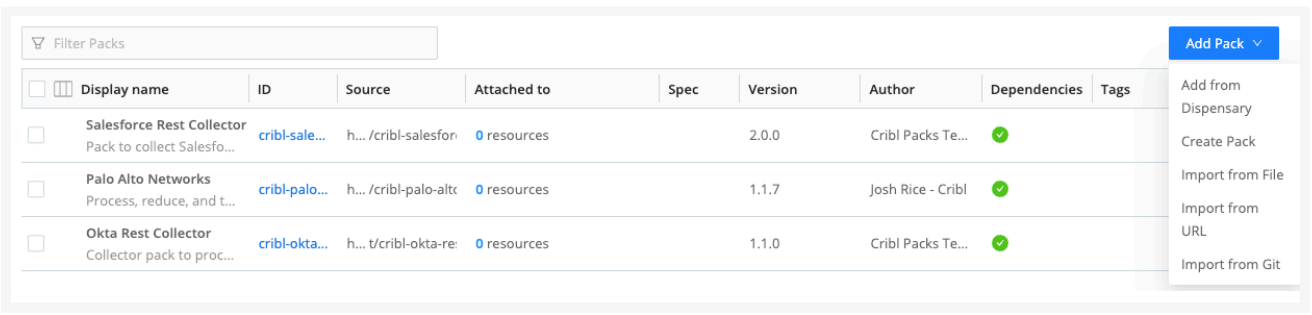
By providing end-to-end configurations for specific services, Packs enable you to:

- **Apply ready-made building blocks:** Install pre-built Packs from the Cribl Packs Dispensary to quickly integrate services and adapt them to your requirements. See Onboard Faster With Packs for information on using Dispensary Packs.
- **Maintain consistent configurations:** Capture organization-specific best practices and share standard workflows across your entire deployment. See Pack-Based Configuration Management to learn how to templatize your end-to-end deployment into a portable, reusable unit.
- **Learn from existing configurations:** Open and review Packs to see how configurations implement reduction, enrichment, normalization, and routing patterns. See Use Packs as Examples.
- **Share and contribute:** Package your own complex workflows to distribute to other teams, or publish them for the wider Cribl community. See the Packs Publication Standards for community guidelines on contributing to the Dispensary.

Get Packs

You can add Packs to your deployment by importing them from:

- Packs Dispensary
- File
- URL
- Git repo



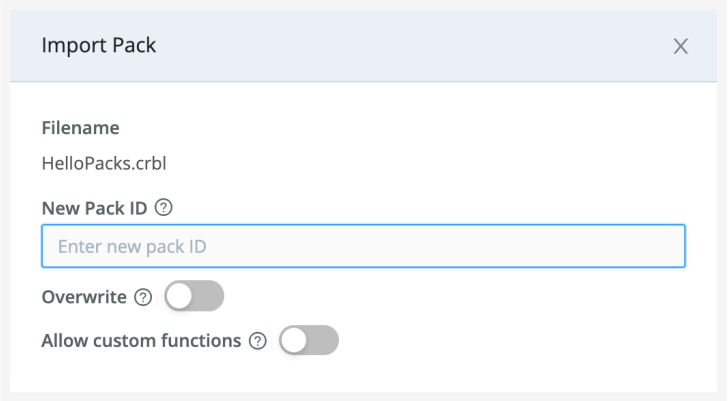
Importing a Pack

Pack Naming on Import

Each Pack that you install within a given Fleet (or Single-instance deployment) must have a unique ID. The ID is based on the internal configuration of the Pack - not its container file name, nor on its **Display name**.

If you import a Pack whose internal ID matches an installed Pack - whether an update, or just a duplicate - you can choose one of two options:

- Assign a unique **New Pack ID**. This will import the Pack as a new, separate Pack with the new ID.
- Toggle **Overwrite** on to overwrite the installed Pack, reusing the same ID.



Renaming a Pack on import

Be careful about enabling the **Overwrite** option. Doing so will cause the imported Pack to completely overwrite your existing Pack configuration.

Each Pack within a Fleet must have a unique **Pack ID**, so you cannot share an ID between two (or more) installed Packs.

Add From Dispensary

Browse and download Packs from the Cribl Packs Dispensary. See Onboard Faster With Packs for more information about using Packs from the Cribl Dispensary.

Import from a File

To import a Pack (`.crbl` file) from your local filesystem:

1. Navigate to the **Packs** page.
2. Select **Add Pack** at the upper right and select **Import from File**.
3. Select the file to import.

4. Optionally, give the Pack an explicit, unique **New Pack ID**. (For details about this option, see Upgrade an Existing Pack.)
5. Select **Import** to confirm the import.
6. If the Pack you are importing binds variables to any fields, you will see a **Configure Variables** button in the notification about successful import. Select it to go to the **Variables** page, where you can configure the values for the variables for your deployment.

Import from a URL

To import a Pack from a known, public or internal, URL:

1. Navigate to the **Packs** page.
2. Select **Add Pack** at the upper right and select **Import from URL**.
3. Enter a valid URL for the Pack's source. (This field's input is validated for URL format, but not for accuracy, before you submit the modal.)
4. Optionally, give the Pack an explicit, unique **New Pack ID**. (See Upgrade an Existing Pack.)
5. Select **Import** to confirm the import.
6. If the Pack you are importing binds variables to any fields, you will see a **Configure Variables** button in the notification about successful import. Select it to go to the **Variables** page, where you can configure the values for the variables for your deployment.

To import a Pack from a public URL, the Leader Node (or single instance) requires internet access. A Distributed deployment Leader can then deploy the Pack to Workers even if the Workers lack internet access.

Import from Git Repos

When you import a Pack from a private Git repository using a personal access token, you can also push changes back to the same branch or repository you imported from. If you don't provide an access token, Cribl Edge cannot write back to the repository. This means you won't be able to push changes to a repository without authenticating with a personal access token.

Some additional considerations:

- If you don't specify a branch during import, Cribl Edge assumes you want to use the repository's default branch.
- You cannot push changes to a Git tag. Importing from a tag is allowed, but writeback is only supported when importing from a branch.

- If you need to change the token used for Git operations, you'll need to re-import the Pack using the new token.

To import a Pack from a known public or private Git repo:

1. Navigate to the **Packs** page.
2. Select **Add Pack** at the upper right and select **Import from Git**.
3. Enter the source repo's valid URL.
   This field's input is validated for URL format, but not for completeness or accuracy, before you submit the modal. When targeting a private repo, use the format: `https://<username>:<token/password>@<repo-address>`. Public repos need only `https://<repo-address>`, as shown in the following example.
4. Optionally, give the Pack an explicit, unique **New Pack ID**. (See Upgrade an Existing Pack.)
5. Optionally, enter a **Branch or tag** to filter the import source using the repo's metadata. You can specify a branch (such as `master`) or a tag (such as a release number: `0.5.1`, and so forth).
6. Select **Import** to confirm the import.
7. If the Pack you are importing binds variables to any fields, you will see a **Configure Variables** button in the notification about successful import. Select it to go to the **Variables** page, where you can configure the values for the variables for your deployment.



To import a Pack from a public repo, the Leader Node (or single instance) requires internet access. A Leader in a Distributed deployment can then deploy the Pack to Workers even if the Workers lack internet access.

Prepare Pack for Use

After you have imported a Pack, take the following steps to ensure it is ready for use on your Edge Nodes:

- **Verify secrets:** If the Pack was exported in `merge` mode, any encrypted fields will have been removed for security. You must re-enter your secrets in the imported Pack to ensure it works as expected.
- **Configure variables:** If you did not configure values for a Pack variable during import, you can do it later by navigating to **Knowledge** > **Variables** in the imported Pack.
- **Commit and Deploy:** If your Pack includes Collectors and jobs, you must Commit and Deploy it after import. This step is critical because it ensures your Edge Nodes have the correct Pack context to successfully run jobs.

Use Packs

Version Compatibility

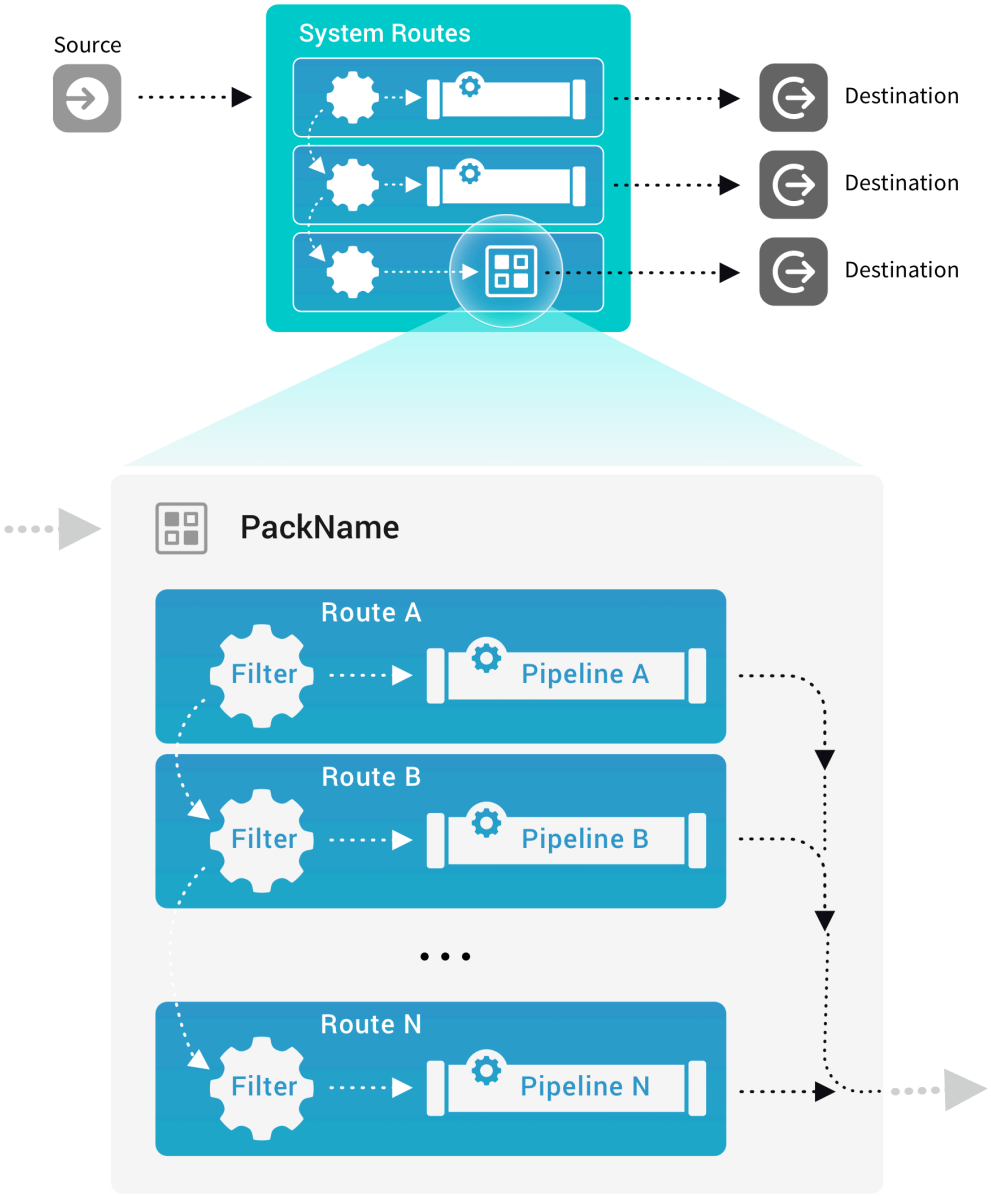
Packs created or modified in Cribl Edge 4.0.x cannot be used in any Cribl Edge version older than 4.0. (If you try, you'll see a `should NOT have additional properties` error.) To avoid this problem, Cribl recommends that you upgrade to Cribl Edge 4.0.x or newer.

For compatibility questions about any individual Packs, please contact us in the Cribl Community Slack `#packs` channel.

For Packs that do not contain Sources or Destinations, you can use them in all places where you can reference a Pipeline:

- In Sources, where you attach pre-processing Pipelines.
- In Destinations, where you attach post-processing Pipelines.
- In Routes, in the Routing table's **Pipeline** and **Destination** columns.

This expanded view shows how a Pack can replace a Pipeline in a Route:



A Pack can play the roles of an enhanced Pipeline

Packs are distinguished in the display with a **PACK** badge, as you can see here in the Routing table:



PACK badge in Routing table's Pipeline column

The **PACK** badge is also displayed when you click into a resource - shown here on one of the Routes from the above table:



PACK badge on a Pack connected to a Route

Cribl Edge's **Monitoring** page includes a **Packs** link where you can monitor the throughput of a Pack.

Access Packs

You access Packs differently, depending on your deployment type.

Single-Instance Deployment

In a Single-instance deployment, Packs are global. On the top bar, select **Processing** > **Packs**. On the filesystem, Packs (including those that you add) are stored at `$CRIBL_HOME/default/`.

Distributed Deployment

In a Distributed deployment, Packs are associated with (and installed within) Fleets. In the sidebar, select **Fleets**, and then select the Fleet you want to manage. Next, from that Fleet' submenu, select **Processing** > **Packs** (Stream) or **More** > **Packs** (Edge).

Each Group's Packs are stored at `$CRIBL_HOME/groups/<groupName>/default/`.

By design, you can readily share Packs **across** Fleets by copying them between Groups.

Use Packs as Examples

In addition to running Packs in production, you can open them to see complete, working configurations:

- How Routes, Pipelines, and knowledge objects are wired together for a specific data source or use case.
- How complex Function chains implement common patterns such as reduction, enrichment, or field normalization.
- How sample data and Data Preview are used inside a Pack to test changes safely.

This makes Packs a useful reference library when you are:

- Learning how to configure a new integration.
- Looking for a proven Pipeline pattern (for example, reducing noisy events or cleaning malformed data).
- Studying how a Pack models a particular vendor, technology, or log format so you can reuse or adapt the approach in your own configuration.

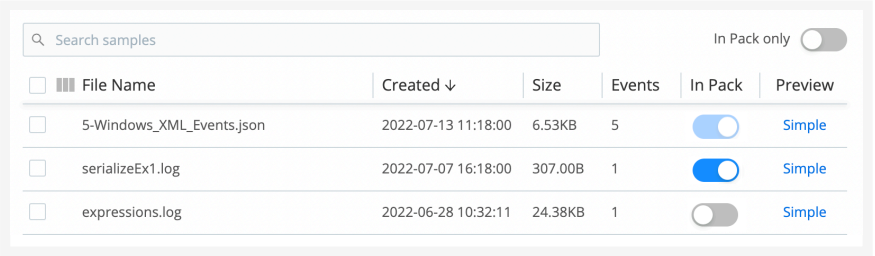
Use Sample Data

Within a Pack, the right pane defaults to displaying all sample data files available on your Cribl Edge instance. If you prefer to filter only sample files internal to the Pack, select the **In Pack only** option.

If you add sample data files via a Pack UI, they will be internal to that Pack. Each sample file here displays its own **In Pack** toggle on its row, which works as follows:

A light-blue toggle is locked, meaning that this sample file is internal to the Pack. It will export with the Pack. If you want to make this sample available across Cribl Edge, you'll need to also add it via the global right preview pane (accessed from **Routing** > **Data Routes** or **Processing** > **Pipelines**).

The toggle indicates whether the sample file is global to Cribl Edge and available to this Pack. Toggle on if you want the sample file to export along with the Pack.

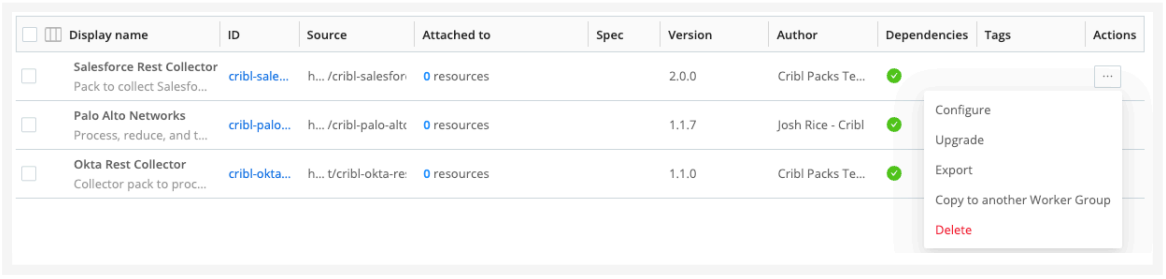


Sample file in Pack

Basically, you can manipulate all the options here as you would work with the equivalent options in the global navigation.

Upgrade an Existing Pack

To upgrade an **existing** Pack, open the Options ••• menu on its row, and then select **Upgrade**.





If you've modified an installed Pack, Cribl Edge will block the overwrite of the Pack, to prevent deletion of your locally created resources.

When upgrading a Pack, Cribl recommends that you:

- Import the updated Pack under a new name that includes the version number (for example, `cribl-syslog-input-120`). This allows you to review and adjust new functionality against currently-deployed configurations.
- Do a side-by-side comparison of the previous and new versions of the Pack - remember to review all comments in the new Pack. Doing this side-by-side comparison allows you to copy Function expressions and other settings from the current version into the same fields in the new version.
- Enable or disable any Functions in the new Pack as necessary.
- Update any Routes, Pipelines, Sources, or Destinations that use the previous Pack version to reference the new Pack.
- If the Pack includes any user-modified versions of default Cribl Edge Knowledge objects (for example, lookup files): Be sure to copy the modified files locally for safekeeping, before upgrading the Pack. After you install the upgrade, copy those files back to the upgraded Pack, overwriting the default versions in the Pack.
- Test, test, and test!
- Commit and Deploy.

Copy a Pack

You can share Packs in a Distributed deployment by copying the Packs between Fleets. You can copy one or more Packs from the **Packs** page as described here or using the Cribl API.

To copy a single Pack, your first step is to open its Options ••• menu, then select **Copy to another Fleet**.

To copy multiple Packs in one operation:

1. Select the check boxes for all Packs you want to copy on the **Packs** page.
2. In the options ••• menu, select **Copy selected Packs to another Fleet**.
3. Select the Fleets to which you want to copy the Pack.
4. Optionally, select the **Overwrite** option to replace existing Packs that have the same **Pack ID**.



Be careful about enabling the **Overwrite** option. Doing so will cause the imported Pack to completely overwrite your existing Pack configuration.

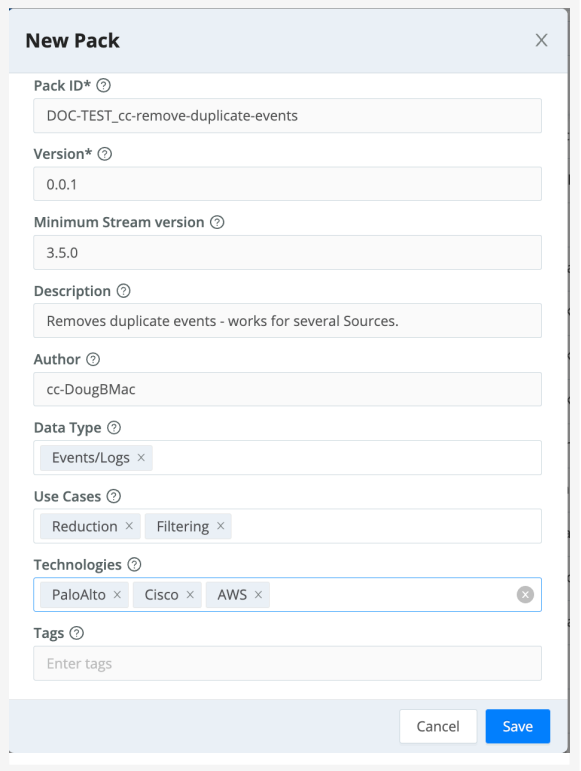
5. Select **Copy**.

A status modal will either display a success message, or list any Packs that failed to copy.

Create a Pack

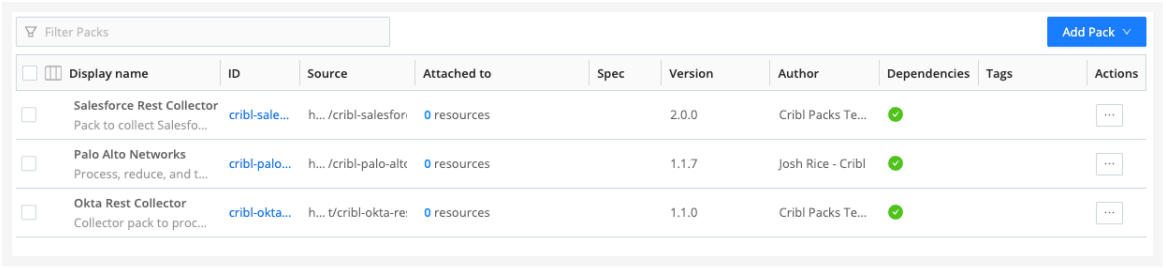
You can create a new Pack from scratch, to consolidate and export multiple Cribl Edge configuration objects:

1. Navigate to the **Packs** page and select **Add Pack**.
2. From the submenu, select **Create Pack**.
3. Enter a unique **Pack ID** and other details:
    - Each Pack within a Fleet must have a separate **Pack ID**, but you can assign arbitrary **Display names**.
    - **Version** is a required field identifying the Pack's own versioning.
    - **Minimum Edge version** is an optional field specifying the lowest compatible version of Cribl Edge software.
    - **Description** and **Author** are optional identifiers.
    - **Data type**, **Use cases**, and **Technologies** are optional combo boxes. You can insert one or multiple keywords to help users filter Packs that you post publicly on the Cribl Packs Dispensary.
    - **Tags** are optional, arbitrary labels that you can use to filter/search and organize Packs.
4. Select **Save**.



Creating a Pack

1. On the **Packs** page, select the new Pack's row to open the Pack.



Manage Packs page

1. Use the standard Cribl Edge controls to configure and save the infrastructure you want to pack up. As you save changes in the UI, they're saved to the Pack.

Pack Variables

To make your Packs portable and reusable across different environments, replace fixed values in your configuration with variables and secrets. This practice, known as templatization, allows your Pack to adapt to different environments without you having to manually edit it each time.

For more information about Pack variables, see:

- Pack Variables
- Create a New Variable
- Use Variables in a Pack
- Update Variables After Import or Update
- Delete Variables

Use Named TLS Certificates

Cribl Edge allows you to optionally authenticate to Sources or Destinations using TLS (Transport Layer Security), which encrypts traffic and helps ensure secure communication with upstream or downstream services. TLS certificates help verify the identity of the remote service and can also support mutual authentication when required. See Secure Cribl Edge Sources and Destinations with Certificates for general information about authenticating with TLS certificates and specific instructions on adding and selecting a certificate to a Source or Destination.

For Sources or Destinations defined at the **global level** (outside of a Pack), Cribl Edge stores TLS certificates at the Fleet level. Any Edge Nodes in the Fleet can access those certificates when processing data.

For Sources or Destinations defined at the **Pack level** (inside a Pack), you can reference a TLS certificate by its name. This approach enables portable and secure configurations across environments:

- When you **export** that Pack, it does not include the TLS certificate itself, only the reference to the certificate name.
- When you **import** that Pack to a Fleet, Cribl Edge checks the local certificate store for that Fleet. If a certificate with the same name exists, Cribl Edge automatically binds it to the Fleet.

This approach allows you to:

- Build TLS-secured Sources and Destinations directly within Packs.
- Reuse Pack configurations across environments without exporting certificate data.
- Maintain secure, environment-specific certificate management practices.



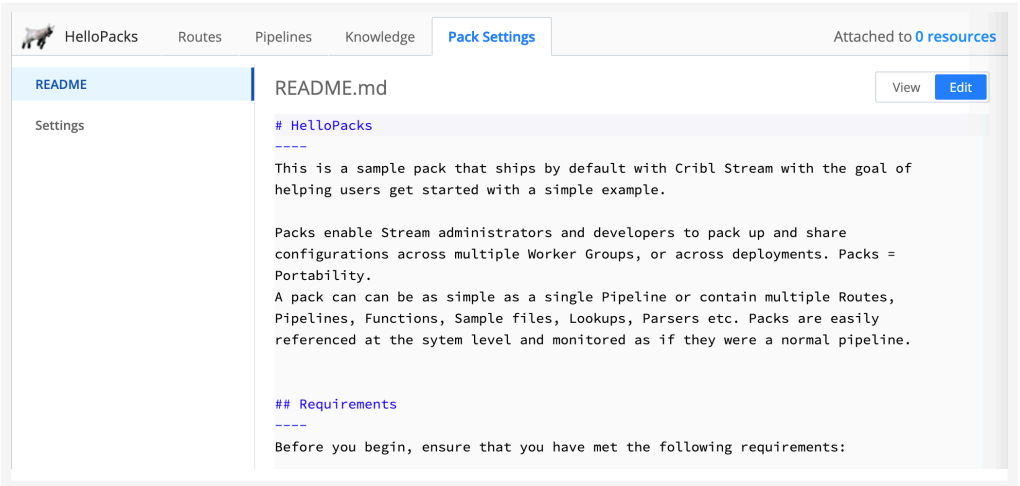
Packs do not include certificate files. Instead, they store the name of the certificate to reference. To ensure a successful deployment, make sure each environment contains a certificate with the expected name for the target Fleet.

If you create the certificate after importing the Pack, commit and deploy the changes to restart the Edge Nodes. The Edge Nodes will pick up the new certificate after restart.

Modify Pack Settings

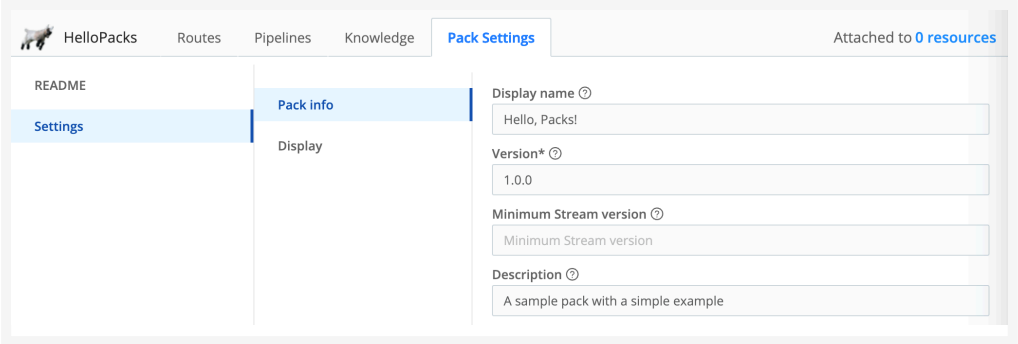
You can update Pack metadata (such as Version, Description, and Author) and the Pack's display settings. If you're developing a new Pack to share, you'll want to use this interface to populate the README and display logo within the Pack:

1. Open a Pack and select **Pack Settings**. The left **README** tab will gain focus.
2. To populate the Pack README file, toggle **View** to **Edit**, replace the placeholder markdown content with your actual Pack description, and **Save**.



Editing a Pack's README

1. To update other metadata, select the left **Settings** tab.



Editing a Pack's metadata

1. You can add a Pack logo on the **Settings** tab.
   Cribl recommends adding a logo to each custom Pack, to visually distinguish the Pack UI from the surrounding Cribl Edge UI (as well as from other Packs). You can upload a `.png` or `.jpg`/`.jpeg` file, up to a maximum size of 2 MB and 350x350 px. Cribl recommends a transparent image, sized approximately 280x50 px.

Share and Publish Packs

To share your Pack with other users, you first need to export it from your deployment.

Then, if you wish, you can publish it to the Cribl Pack Dispensary to make it available for the whole Cribl ecosystem.

Export a Pack

You can also use the Cribl API to export a Pack.

To export a newly created or modified Pack from the **Packs** page, open its Options ••• menu and select **Export**.

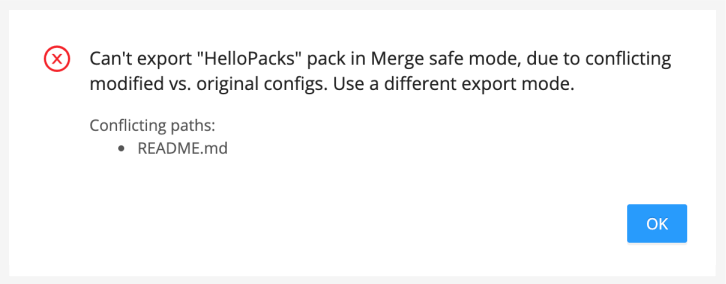
The **Export Pack** modal provides the following options.

Export Mode

Select one of these three buttons to determine the export mode:

- **Merge safe**: (Deprecated) Attempt to safely merge local modifications into the Pack's default layer (original configuration), then export.
- **Merge**: Force-merge local modifications into the Pack's original configuration, then export. When you export Packs in `merge` mode, Cribl Edge deletes all encrypted fields from Pipelines. Upon importing these packs, you will need to re-enter the secrets to make them functional again.
- **Default only**: Export only the Pack's original configuration, without local modifications.

The (deprecated) **Merge safe** option is conservative, and will block the export where Cribl Edge can't readily merge modified contents with the Pack's original contents, due to conflicts. If you encounter an error like the example shown below, use the **Merge** or **Default only** export mode instead.



Merge safe error

Exported Pack ID

Defaults to the Pack's current ID, with the version number appended. This field provides an opportunity to change the Pack's ID, if you're exporting a new version of the Pack.

Publish a Pack

If you'd like to share your Pack with the community of Cribl users, you can publish it on the Cribl Packs Dispensary.

The Cribl Packs Dispensary site is designed for sharing completed Packs. If you want to collaborate with others on iteratively developing a Pack, Cribl recommends relying on our Dispensary GitHub Repo for the development phase.

If you have a Cribl.Cloud account, you can also collaborate there by inviting team members to your Cribl.Cloud Organization. See Invite Members.

Once your Pack is ready to share, we encourage you to submit it to the Cribl Packs Dispensary site. If you already have completed Packs on our GitHub repo, bring them over here!

Change Pack Owner

If you no longer can or wish to maintain a Pack you published to the Cribl Packs Dispensary, you can transfer its ownership to another user. The new owner will have full control over the Pack.

To transfer a Pack to a new owner:

1. Log in to the Cribl Packs Dispensary.
2. Optionally, toggle **View only my Packs** to **Yes** to narrow down the list to the Packs you own.
3. Select **Configure** on your Pack's tile.

4. At the bottom of the Pack pane, select **Change Owner**.
5. Enter the email address of the new owner and, optionally, a message you want to send them.
6. Confirm by selecting **Change Owner**.

The recipient will get an email message with a request to accept ownership of the Pack. They will have 24 hours

16.1. Pack-Based Configuration Management

As your Cribl deployment grows, maintaining consistency across multiple Fleets and environments becomes a significant operational challenge. To solve this challenge, use Pack-based configuration management to templatize your entire end-to-end Cribl data streams into reusable units that you can port to other Fleets in your deployment. This approach is helpful when you need standardized configuration management workflows to manage multiple Fleets, instances, or environments (such as development, testing, and production).

Pack-based configuration management has several advantages over the standard Leader-to-Fleet configuration management workflow:

- **Composable management:** Define end-to-end data streams as a single unit. Package the configurations for a Source, its associated processing logic, and the final Destination into a reusable Pack that you can deploy across Cribl Edge to different Fleets and environments.
- **Git version-control:** You can version-control Packs in an external Git repository to make your Cribl configurations auditable and to flexibly adapt to your company's change management policies.
- **Environmental portability:** Move configurations easily between environments, such as from `dev` to `prod` or more. Packs allow you to bind configuration fields to variables, so that you can apply the same configuration components to multiple Fleets and then customize the configuration details that are specific to the local context of a given Fleet.
- **Scalability:** Onboard new data streams or spin up additional Fleets quickly by deploying Packs that have been validated in one context to another.

When To Use Pack-Based Configuration Management

Pack-based configuration management is not required to manage deployments of Cribl Edge. For many deployments, the default Leader-to-Worker configuration management system is sufficient.

However, this workflow is particularly beneficial for organizations that require:

- **The ability for multiple configuration content developers or teams to collaborate on configurations.** Use this workflow if you need developers to work on configurations in parallel. Each content developer can work on isolated feature branches, merge changes systematically, and resolve conflicts before they impact production.
- **Two- or three-branch environment promotion and segregation workflows.** This workflow supports the classic development three-branch model where changes flow from `dev` to `test` to `prod` branches and environments in a controlled manner. You could also use a simpler two-branch model flow from `dev` to `prod` instead.

- **Formal configuration approval from a central Change Advisory Board (CAB).** Git workflows keep configuration changes visible so that stakeholders can review and approve them. An external Git repository can provide additional change traceability and a clear audit trail of who approved what and when.
- **The ability to roll back configurations quickly.** While this is already possible in Cribl Edge, the Pack-based configuration management workflow extends and complements the Leader-to-Fleet control plane system by allowing you to track changes in an external Git repository. Git's version control capabilities make rollbacks straightforward so that you can easily revert to a previous, stable version. Connecting to an external repository provides additional visibility and auditability that integrates with your company's overall change management system.

Prior experience with Git is helpful for this workflow. For users familiar with Git, this workflow can streamline version control, change management, and CI/CD for your Cribl environments.

Key Components

The recommended Pack-based configuration management workflow for Cribl Edge uses these components:

- Packs: Packs contain the actual configuration content that you will track. This includes dependencies such as the Knowledge objects or (optionally) the sensitive resources (such as certificates or secrets) needed to run the Pack.
- Environments: A logical grouping of Cribl Edge resources that process data, depending on your Fleet placement strategy. An environment is typically defined by a specific set of Fleets.
- Variables: Dynamic placeholders within a Pack configuration that map to a specific values defined at either the Pack or Fleet level. Variables allow you to port the Pack to another environment, then make small adjustments to the logic to allow for specific differences that are unique to that Fleet.
- External Git repository: A centralized Git repository serves as the single source of truth, tracking all changes to your Pack configuration. Each repository should track one individual Pack only.

Packs

The core building block of every Cribl Edge configuration management system is the Pack. A Pack contains a bundle of configurations that you want to develop, test, version control, and distribute to multiple Organizations or Fleets. Packs can contain configurations for many types of resources:

- Sources
- Destinations
- Event Breakers
- Pipelines
- Routes

- Knowledge Objects (such as lookup files)
- Sample data files

In other words, you can create Packs that contain a full, end-to-end Cribl Edge configuration. Think of a Pack as a configuration object that you can version control and export/import to other environments. It works well in workflows where multiple developers work in a sandbox development environment. Developers can review each other's work and merge their changes to a testing or production environment using Git workflows.

Environments

An Environment is a logical boundary that defines how data is processed, managed, and secured within your architecture. The logical boundary is defined by your Fleet placement strategy, such as geographic regions or data proximity. It could also refer to content development environments that use the two- or three-branch development strategy, such as `dev`, `staging`, and `prod`.

While typically defined by a specific set of Fleets, an environment in the Pack-based workflow encompasses the unique variables and configurations required for that specific environment. This means a single Pack can be deployed across multiple environments while automatically adapting to the new context.

Variables

Variables allow you to copy or import that Pack to multiple Fleets, making small adjustments to the logic to adapt to the specific environment context for that Fleet. For example, you could adjust the Fleet-level variables that account for regional differences (such as different AWS regions), VPC endpoints, or local port assignments. This ensures that the core data processing business logic remains intact while adapting to the local requirements of its new Fleet context. It also allows you to easily make updates to Packs and deploy those updates with minimal overhead.

External Git Repositories

Once you've built an end-to-end configuration of Cribl Edge inside a Pack, you can then integrate it with an external Git repository for version control, parallel development, and automated deployment with CI/CD.

Cribl Edge is compatible with all major Git platforms, including GitHub, GitLab, and Bitbucket.

Additional Configuration Management Components

This table explains how standard configuration management components map to Cribl Edge:

Element	Explanation	How it Applies in Cribl
Automation agent or runner	A process or tool that pulls from Git and applies the config to the target environment.	Handled inside Cribl Edge by committing and deploying imported Packs to Fleets. Can also be handled using a CI/CD pipeline tool in external repositories, including GitHub Actions, GitLab CI, and Jenkins.
Change workflow	Configuration changes go through a review and approval process.	Handled in an external Git repository linked to Cribl Edge in compliance with your internal change management policies.
Target system(s)	Where the configuration is ultimately applied.	Instances of Cribl Edge.
Secret management	Secrets or encrypted files.	Handled inside Cribl Edge and mapped to variables in external Git repositories.
Environment mapping	Directories or branches per environment.	Handled with variables inside Cribl Edge and mapped to variables in external Git repositories.

Considerations

When working with Packs, keep these considerations in mind:

- Packs cannot be cloned or imported from one Cribl product to another. For example, you cannot clone a Pack from Cribl Stream to Cribl Edge and vice versa.
- Sources and Destinations created in Packs add the Pack name to the `__inputId` and `__outputId` fields of their events. For example: `datagen:My_Pack.my_datagen_source`, or `cribl_http:My_Pack.my_http_destination`.

16.1.1. Plan A Pack-Based Strategy

To get started with Pack-based configuration management for Cribl Edge, your first objective is to create a Pack-based configuration management strategy that fits your team's workflows and operational needs. A thoughtful plan helps reduce duplication, simplify maintenance, and make your deployments more predictable and secure.

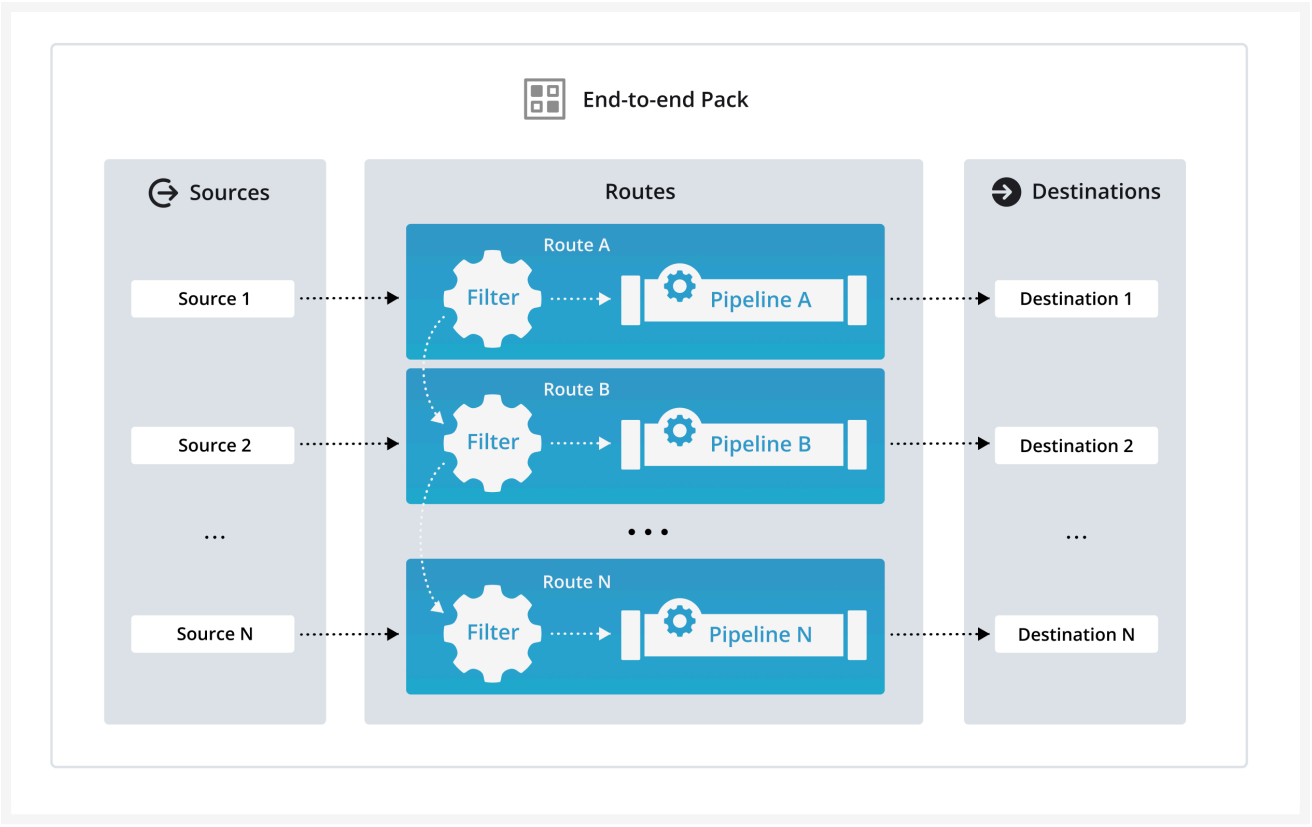
Your configuration plan should define:

- The Purpose and Scope of the Pack: The scope of the configurations in your Pack, which involves deciding whether to include Sources and Destinations. It also includes determining how data will flow in and out of Packs to other resources in Cribl Edge.
- Pack Variables: The variables you want to templatize in your Pack for easy transfer and adaptation across environments.
- Sensitive Data and Secrets: The sensitive data you need to store in Packs, including passwords, API keys, or security certificates, and authentication credentials.
- Pack Versioning System: Your Pack versioning and branch promotion strategies, which ensures your Packs remain consistent and synchronized across your Fleets.

Purpose and Scope of the Pack

The core building block of every configuration management system is the Pack. A Pack contains a bundle of configurations that you want to develop, test, version control, and distribute to multiple Organizations or Fleets.

If you're familiar with Packs from prior versions of Cribl Edge before 4.12, it's important to know that Packs have evolved. In previous versions, Packs were mainly used to package processing logic and included Pipelines, Routes, Functions, Knowledge objects, and sample data files. Packs can now contain complete, end-to-end data flow configurations for a single data stream.



Packs can contain end-to-end data flow components

For that reason, carefully plan the scope of the Pack you want to track with version control, which primarily means deciding whether to include Sources and Destinations in your Pack:

Scope	When to Use	Contains
End-to-end scope	When you want to configure your full end-to-end Cribl Edge deployment inside a Pack.	• A Source. • A Destination. • All the necessary components to process data from that Source to that Destination (Pipelines, Routes, and related Knowledge objects needed for that configuration to work).
Source-side scope	When you want to use a Pack similar to a pre-processing Pipeline.	• A Source. • Optionally, a Pipeline and all the necessary components to normalize, reduce, or process data from that Source, including Knowledge objects.
Destination-side scope	When you want to use a Pack similar to a post-processing Pipeline.	• A Destination.

Scope	When to Use	Contains
		• Optionally, a Pipeline and all the necessary components to prepare or optimize the incoming data for the requirements of the downstream receiver.
Data processing scope	When you only want the Pack to process data, meaning it will just contain a Route and Pipeline and related Knowledge objects.	• Only include Routes and Pipelines that process data. • Do not include Sources and Destinations in these Packs.

Keep each Pack scoped to a single purpose or use case where possible. Avoid mixing too many unrelated Sources or Destinations in one Pack. It increases complexity and makes version control more difficult. Consider these possible approaches for designing your Pack scope:

- **One Pack per vendor or technology:** Create a Pack for a specific technology (such as AWS CloudWatch or Syslog messages).
- **One Pack per data lifecycle stage:** Divide Packs by their function: one for normalization, one for enrichment, and one for redaction.
- **Packs for shared services:** Create a Pack that performs a universal task (such as redaction) that multiple other data streams can be routed through before reaching their next Pipeline or Destination.

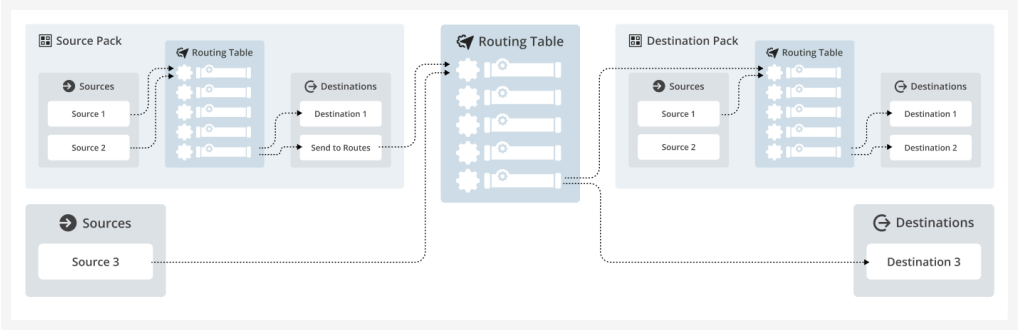


Packs are fully self-contained, meaning that the objects inside the Pack are not aware of anything outside of the Pack context unless you use Routes to send data outside the Pack context. So, if you include a Source or Destination inside a Pack, **it won't be aware of data from an external Source or Destination** unless you integrate it into your data processing system by connecting it to a Source or Destination.

The rest of this guide assumes you plan to build Packs that include Sources and Destinations. However, you can adapt this guide to Packs with more limited scopes.

Design the Pack Data Flow

Packs allow you to break down repetitive configurations into modular, reusable components that can be chained together into a single data stream flow. Rather than a single, complex Pipeline at the Fleet level, you can chain specialized Packs together by routing data through Routes at the Pack level.



Route data in and out of Packs

You can route data at two different scopes:

- **At the global Fleet level:** You can route data to or from Packs using the global Routing table at the Fleet level.
- **At the Pack level:** You have a self-contained Pack that contains a Source or a Destination (or both). Data can be processed in the Pack and then routed as it leaves the Pack as needed.

If you include a Source, Collector, or Destination in your Pack, that Pack will not be available when you select the **Pipeline** menu in Routes at the global Fleet level. Only Packs scoped exclusively for data processing appear in the **Pipeline** dropdown. See Purpose and Scope of the Pack for a deeper explanation of how the content of Packs can be scoped.

You can design the scope of Packs using a few patterns that consider the overall flow of data:

- **Source-side Packs:** These Packs can contain one or more Sources and any specific Pipelines needed to normalize raw data into a standard internal schema. You could also use this Pack to drop or aggregate any unnecessary fields and events to optimize incoming data. From there, the data can flow out of the Pack to Routes at the Fleet level for further processing.
- **Destination-side Packs:** These Packs can contain one or more Destinations and any specific Pipelines that prepare or optimize the incoming data for the requirements of the downstream receiver. The Pipelines in these Packs can refine, format, and batch data so that the Destination can ingest it as efficiently and cheaply as possible. Routes are the mechanism that sends data from the Fleet level to these Packs.
- **Pass-through processing Packs:** Data originates from a Source that exists at the Fleet level, then passes through Routes, which sends the data to a Pack for specific enrichment step before returning to Routes for further routing.

These workflows are ideal for organizations that require functional decoupling of their configuration logic. By segregating Source and Destination logic into independent Packs, individual contributors can iterate on specific data streams in isolation, eliminating the risk of cross-contamination or breaking global configurations.

Avoid processing data twice. When planning your flow, decide exactly where heavy data processing (like masking or Regex operations) should occur. Performing the same transformation in both a Source-side Pack and a Destination-side Pack wastes CPU cycles and increases latency.

See Connect Packs to Global Sources and Destinations for more specific information about how to route data in and out of Packs.

Pack Variables

If you plan to use the same Pack across multiple environments, take some time to identify which configuration values are environment-specific. These may include:

- Endpoint URLs or hostnames.
- Port numbers.
- Region-specific settings.
- Authentication details.

To reduce duplication and make your Packs reusable across environments, use Pack variables to templatize these values. In Cribl Edge, Pack variables let you define values that can be adapted to the local environment. This allows you to:

- Set environment-relative defaults (such as `api_base_url` or `destination_bucket`).
- Avoid hardcoding environment-specific information directly in your configuration.

Keep the following guidelines in mind about variables:

- **Field binding behavior:** Pack variables are bound to specific fields. If you update the variable's value, the corresponding field will reflect the change. However, if you delete the variable, the field retains its last set value rather than reverting or clearing it. See How Field Binding Works for more detailed information.
- **Field compatibility:** Variables are only supported in certain fields, which means you can't use variables universally in every field. Fields that can accept variables have a **Variable** icon ◈ next to them.
- **Exporting secrets:** Sensitive values, such as passwords or API keys, are bundled up with a Pack and exported along with the Pack.
- **Use variables and naming conventions consistently:** When Packs include Sources and Destinations, use clear, environment-aware naming (such as `src_aws_logs_dev`, `dst_s3_prod`). Consider using Pack-level variables to template environment-specific values.

See Configure Variables and Secrets for more specific information about how to use variables.

How Field Binding Works

When you bind a Pack variable to a configuration field, you create a dynamic link. Updating the variable's value automatically pushes that change to all bound fields. However, if you delete the variable, the binding is removed and the field retains its last resolved value as a static, hard-coded setting.

To see how Cribl Edge manages this binding, select **Manage as JSON** on a Source or Destination that uses a variable. Notice that any field bound to a variable generates two distinct entries that share related names:

Field type	Description	Examples

Static field	A hard-coded value that uses the last known value. This serves as the fallback value in case of deletion.	`"host": "localhost"`
		
		`"udpPort": 3000`

Variable template field	Prefixed by `__template_`, this lists the field name and the stored variable reference.	`"__template_host": "my-host-variable"`
		
		`"__template_udpPort": "my-port-variable"`

The `__template_` entry takes precedence over the static field. If you delete the variable, Cribl Edge removes the `__template_` entry from the JSON. Because the static field still exists, the configuration reverts to the last set value, ensuring the data stream remains functional even though it is no longer dynamic.



Never delete the static version of a configuration field from the JSON, even if a `__template_` field is present. The static field is a required component of the configuration file stored on the disk.

Sensitive Data and Secrets

Cribl Edge allows you to optionally authenticate to Sources or Destinations using TLS (Transport Layer Security), which encrypts traffic and helps ensure secure communication with upstream or downstream services. TLS certificates help verify the identity of the remote service and can also support mutual authentication when required.

When working with Packs, use the built-in secret store and named TLS certificates to manage all sensitive data. These methods are crucial for making your Packs portable and secure across different environments. You should plan to use the same process for all secrets, whether they are passwords, API keys, or security certificates.

Key considerations for your plan:

- **Secret isolation and referencing:** Unlike other configurations, sensitive values and secrets (such as TLS certificates and authentication credentials) are not bundled within Packs when you export them. Instead, a Pack only stores a reference to a secret by its name. This means:
    - You can build TLS-secured Sources and Destinations directly into your Packs.
    - The sensitive data (such as certificates) remains isolated in each Fleet's local secret store.
- **Environment-specific secret management:** Plan to manage unique secrets for each environment (such as `dev` or `prod`) while using the same underlying Pack configuration across all environments.
- **Deployment prerequisites:** Cribl Edge stores secrets and certificates at the Fleet level, so ensure that the secrets and certificates are properly deployed to the correct Fleets. For a Pack to deploy successfully with TLS-secured components, a certificate with the matching name must already exist in the target Fleet's secret store.



Use the Cribl secret store whenever possible. This ensures different Cribl environments can hold their own distinct secret values, without needing you to redefine the setting multiple times within the Pack.

For Sources or Destinations defined at the **global level** (outside of a Pack), Cribl Edge stores TLS certificates at the Fleet level. Any Edge Nodes in the Fleet can access those certificates when processing data.

For Sources or Destinations defined at the **Pack level** (inside a Pack), you can reference a TLS certificate by its name. This approach enables portable and secure configurations across environments:

- When you **export** that Pack, you can choose whether to bundle sensitive resources along with the Pack. If you choose not to bundle the resources, it does not include the TLS certificate itself, only the reference to the certificate name.
- When you **import** that Pack to a Fleet, you can choose whether to overwrite the local certificate store or not. If you choose not to overwrite it, Cribl Edge checks the local certificate store for that Fleet. If a certificate with the same name exists, Cribl Edge automatically binds it to the Fleet.

This approach allows you to:

- Build TLS-secured Sources and Destinations directly within Packs.
- Reuse Pack configurations across environments without exporting certificate data.
- Maintain secure, environment-specific certificate management practices.

Pack Versioning System

The last thing to consider as part of your configuration management plan is what your strategy for updating Packs will be.

Key considerations:

- Packs support semantic versioning structures (`0.0.0`), which correlates to `Major.Minor.Patch`.
- The core principle is that each increment of the Pack's semantic version number signifies a stable, tested state of the configuration suitable for promotion to the next environment.
- You can update Packs to new versions using the **Upgrade** option.
- When you upgrade a Pack, Cribl Edge compares the current Pack version against the Pack version in the external Git repository. If the version number has increased, Cribl Edge imports the latest changes from the Git repository to the Pack in the local environment.

> **IMPORTANT: Commit and Push Changes to Avoid Data Loss**
>
> When you upgrade a Pack with a higher version number, it overwrites any local changes to the Pack. To prevent loss, individual Pack developers should always ensure that they have saved, committed, and pushed changes from their local sandbox environment to their feature branch on the external Git repository before upgrading.

With these considerations in mind, the next section describes a recommended version increment strategy across environments.

Versioning Best Practices

16.1.2. Create A Pack

Now that you have a clear plan for the Cribl Edge elements you want to version control, you can start building an initial Pack to test your basic configuration management workflow.

You can create a new Pack from scratch or you can create a Pack, then copy any configurations you developed in your Cribl Edge deployment into the Pack.

Best Practices for Working With Packs

When creating and working with Packs, keep these considerations in mind:

- **Keep the Pack simple at first.** For the purposes of setting up the initial Pack-based workflow, make your initial Pack simple to start and then test the version control workflow by adding more configurations in later steps.
- **Consider using Pack logos to help you remember the Pack context.** You might understandably lose track of whether you are creating Cribl Edge configurations in a local Pack context or a global context. To give yourself a visual cue that you are working in a local Pack, consider adding a logo to your Pack in the Pack settings. The logo will show up in the top breadcrumb bar so that you can clearly see when you are working in a local Pack context. When you don't see that logo, that will be your cue that you are working in a global context outside of the Pack.

Create a New Pack

Start by creating a new Pack based on the scope and goals you defined in your configuration plan:

1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet.
2. Navigate to **Processing**, then **Packs** and select **Add Pack**. From the submenu, select **Create Pack**.
3. Enter a unique **Pack ID** and other details:
    - **Pack ID:** Required. Each Pack within a Fleet must have a unique alphanumeric **Pack ID**. Some characters like hyphens or underscores are allowed, but spaces are not.

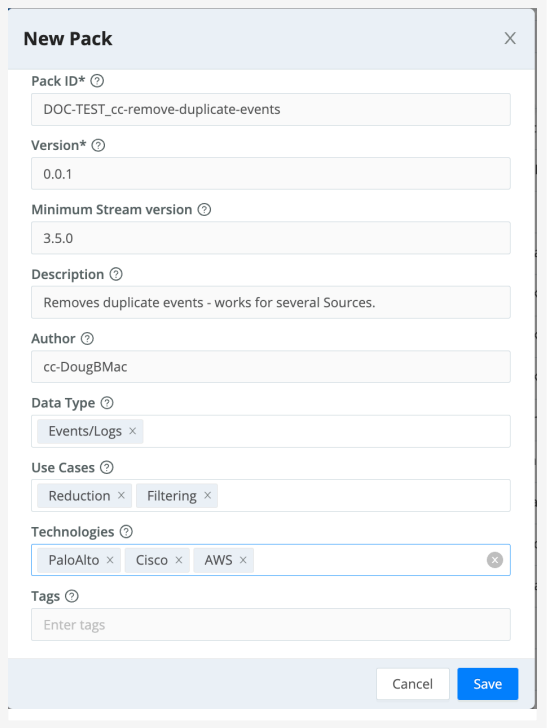
      Pack IDs must be all lowercase. If you enter uppercase letters when creating a Pack ID (such as `MyPack`), Cribl Edge automatically converts the ID to lowercase (`mypack`).

    - **Version:** Required. Set the initial Pack version. The default is `0.0.1`.
    - **Minimum Edge version:** Specify the lowest compatible version of Cribl Edge software. For example, if the Pack uses a feature that was introduced in a specific release, set the version to that release

number.

- **Description:** Add an optional description of the Pack's purpose or contents.
- **Author:** Optionally indicate the author of the Pack.
- **Data type**, **Use cases**, and **Technologies** are optional combo boxes. You can insert one or multiple keywords to help users filter Packs that you post publicly on the Cribl Packs Dispensary.
- **Tags** are optional labels that you can use to filter, search, and organize Packs.

4. Select **Save**.



Creating a Pack

5. On the **Packs** page, select the new Pack's row to open the Pack.



Manage Packs page

6. Use the standard Cribl Edge controls to configure the elements you want to include in the Pack. As you save changes in the UI, they're saved to the Pack.

Copy Resources Into the Pack (Optional)

If you developed configurations outside of a Pack context, you can manually copy configurations from one part of Cribl Edge into a Pack. This section provides steps for copying various resources into your newly created Pack. Select the tab for the type of resource you want to copy over:

Pipelines | Routes | Sources | Destinations

To copy a Pipeline into a Pack:

1. In the sidebar, select **Fleets**, then select a Fleet.
2. On the **Fleets** submenu, select **More**, then **Pipelines**.
3. Locate the Pipeline you want to copy and open the Options ••• menu on its row
4. Select **Copy**.
5. Navigate to your new Pack and, in the Pack submenu, select **Pipelines**.
6. Select **Paste Pipeline** on the filter bar, confirm the Pipeline name and select **Save**.

Configure Variables and Secrets

To make your Packs portable and reusable across different environments, replace fixed values in your configuration with variables and secrets. This practice, known as templatization, allows your Pack to adapt to different environments without you having to manually edit it each time. See Pack Variables and Sensitive Data and Secrets for a reminder about how variables and secrets work.

As you build the Pack, design your variables with their ultimate scope in mind. The portability of Packs comes from the relationship between the Pack itself and the Fleet context. This hierarchy allows a Pack to be environmentally aware, meaning it knows where it is and which local values it should use.

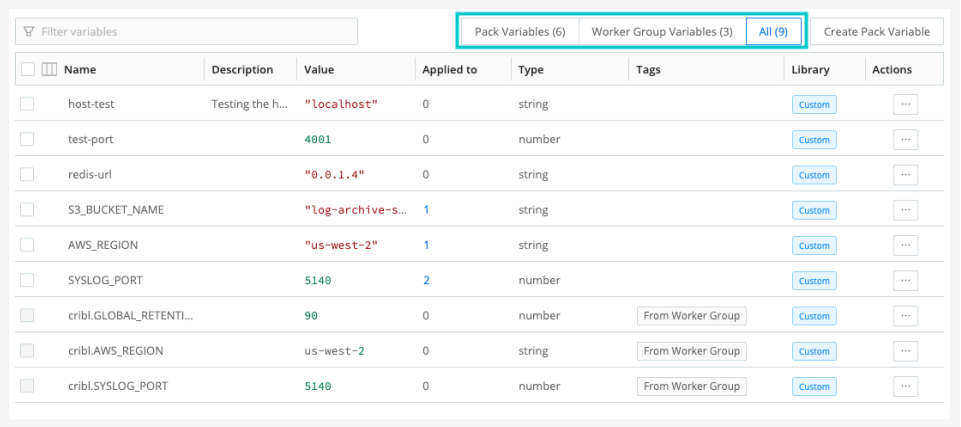
Variable	Description and Use Case	Behavior
Pack	Bundled directly within the Pack. These are best used for internal logic that should remain constant no matter where the Pack is deployed.	When you copy the Pack, these variables move with it automatically.
Fleet	Allow you to define regional or environment-specific values that the Pack inherits once it is installed. For example, AWS regions (us-west-2 vs. us-east-1), VPC endpoints, or local port assignments.	When you port a Pack into a new Fleet, the Pack's configuration fields use the Fleet's context to populate these values.

When you reference a Fleet variable in a Pack, Cribl adds a prefix to indicate that it is a Fleet variable. In Stream, these variables are prefixed with `cribl.`. In Edge, they are prefixed with `edge.`.

Create a New Variable

To create a new variable:

1. From inside the Pack context, select **Knowledge**, then **Variables**. The **Variables** UI has tabs that allow you to select the correct scope for your variable. Select either **Pack Variables** or **Fleet Variables** depending on the desired variable scope.



Select the correct tab

2. Select **Create Pack Variable** and configure the variable settings:
    - **Name:** Enter a unique name for the variable.
    - **Description**: Optionally, enter a description.
    - **Type:** Select the type of variable you want to create. See the Variables Library reference page for a detailed explanation of variable types.
    - **Value:** Enter the value, data structure, or JavaScript expression for this variable. This input must match the format required by the selected variable type. For example, Expressions should have JavaScript syntax. The system validates the input when you save.
    - **Tags:** Optionally, add tags that you can use for filtering and grouping in the variable list view. Use a tab or hard return between (arbitrary) tag names.
3. Save the variable. The system validates the **Value** field to ensure it matches the specified **Type** when it saves.

You can also add new variables directly in the field where you want to reference that variable.

Use Variables in a Pack

Fields that can accept variables have a **Variable** icon ❖ next to them. Select this icon to open the **Add Variable** modal. You can select an existing variable from this list or create a new one.

Fleet variables are designed specifically for configuration binding. They are not accessible using the C.vars expression in Pipeline Functions. You can only bind these variables to Source and Destination configuration fields that display the **Variable** icon ◈.

The only exception is the Redis Function, which can resolve Fleet variables at runtime to manage external state or lookups.

Delete Variables

Before deleting a variable, identify where it is currently applied. When you delete a variable that is bound to a Source or Destination, the system performs the following actions:

- **Converts to static values:** The system replaces the variable reference with its last known value. For example, if a field was bound to an `AWS_REGION` variable with a value of `"us-west-2"`, the field will now contain the hard-coded string `"us-west-2"`.
- **Breaks environmental awareness:** Because the value is now hard-coded, the configuration will no longer adapt if you move the Pack to a different Fleet or update the Pack or variable at the Fleet context.
- **Prevents batch updates:** You can no longer update multiple Sources or Destinations by changing a single variable. You must manually edit each component to update its settings.

This action is permanent and cannot be undone. To restore variable binding, you must manually re-bind each field to a new variable.

Copy a Pack

After building and validating Packs, you can then copy or import that Pack to multiple Fleets, making small adjustments to the logic to allow for specific differences that are unique to that Fleet. For example, you could adjust the Fleet-level variables that account for regional differences (such as different AWS regions), VPC endpoints, or local port assignments.

When you move a Pack, it carries its entire data processing ecosystem with it, including Pipelines, Knowledge objects, and sensitive resources. This ensures that the core data processing business logic remains intact while adapting to the local requirements of its new Fleet context. It also allows you to easily make updates to Packs and deploy those updates with minimal overhead.

To copy a Pack:

1. In Cribl Edge, navigate to the **Packs** page and open the Options ••• menu on the row of your Pack.
2. Select **Copy to another Fleet**.
3. In the modal, define the destination and content:

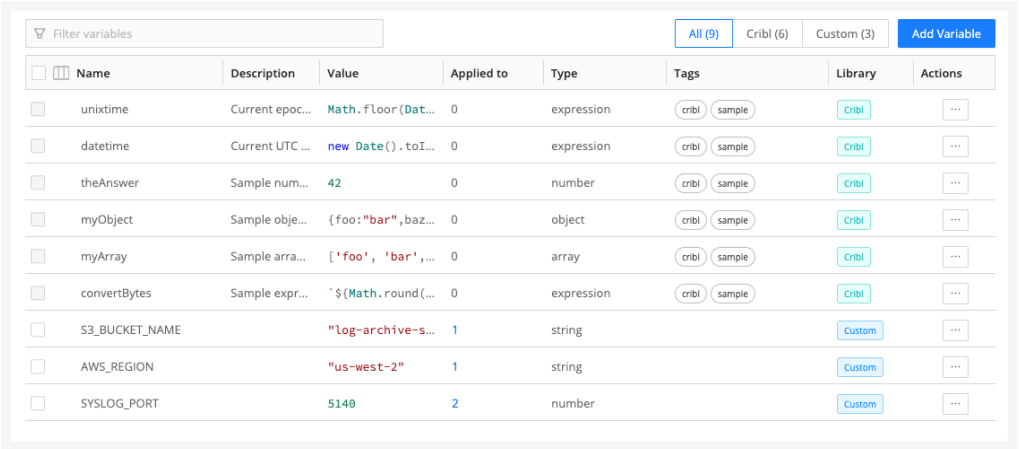
- **Export mode:** Select one of the three available export modes. See Export Mode for more detailed information about each mode.
- **Fleet:** Type or select the target Fleet.
- **Overwrite:** Select this option if you want to replace all configurations, Pipelines, and variables in the target Fleet with the versions from this Pack. Enabling this will permanently overwrite any local modifications made to the Pack in the target Fleet.
- **Include external dependencies (certificates, secrets, variables):** Bundle all certificates, secrets, and variables required for the Pack to function. This ensures the target Fleet has the necessary credentials to run the Pack immediately. Existing resources in the target group are maintained, while only new or missing dependencies are added.

4. Select **Copy**.

Update Variables After Import or Update

When you import or copy a Pack into a new Fleet, Cribl prompts you to configure the variables for that Fleet context by displaying a message that indicates the import was successful. Select **Configure Variables** in this message to go to the **Variables** list. From here, you can see the list of variables available to the Pack, including its Pack variables. You can then update the variables to their custom value for that Fleet.

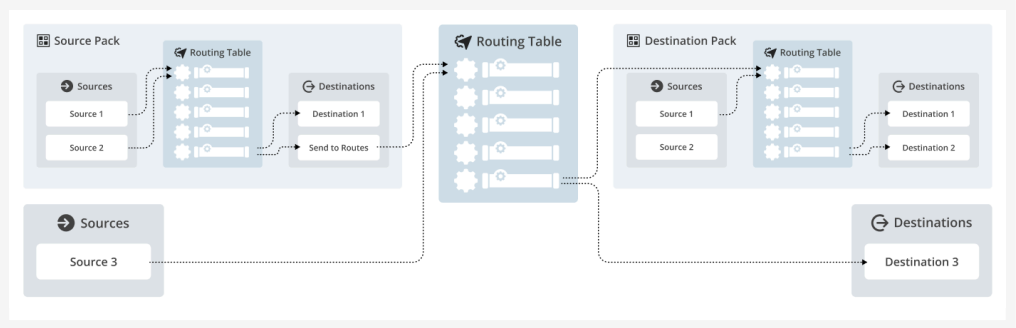
The **Applied to** column displays the number of active references for each variable. Select the value to open a modal listing all Sources and Destinations that reference this variable.



View all Pack variables

Connect Packs to Global Sources and Destinations

After you export a Pack to a new Fleet, you can optionally set up the ability to transfer data in and out of Packs to other Sources and Destinations that are available at the Fleet level. See Design the Pack Data Flow for an explanation of how to plan an effective data flow strategy.

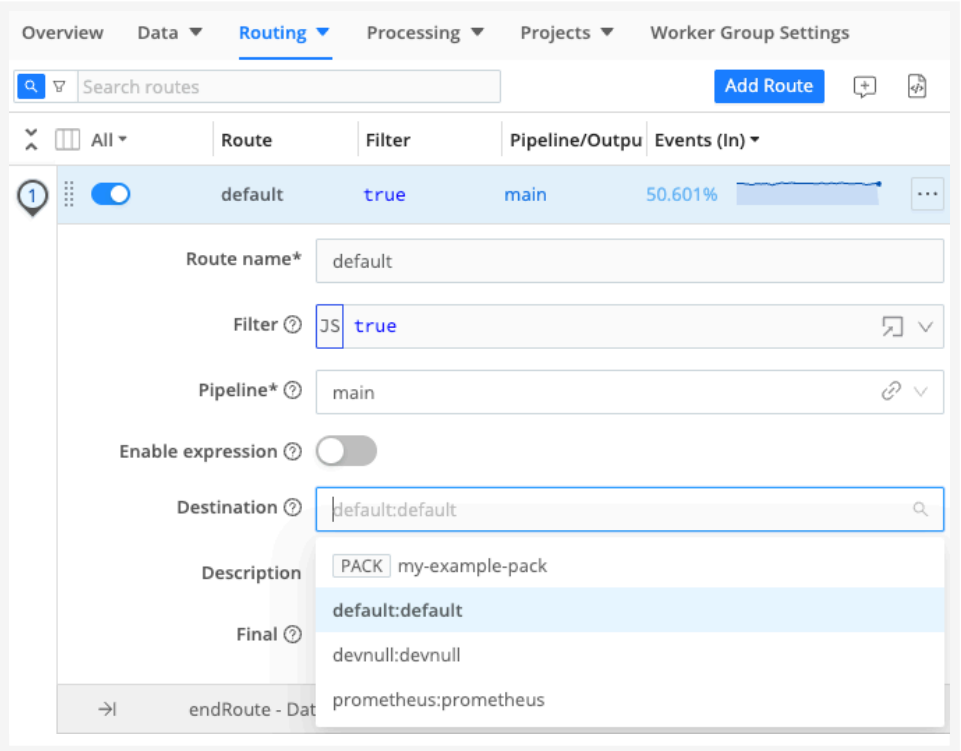


Route data in and out of Packs

Send Data From a Global Source to a Pack

To route data from a Source that exists at the Fleet level to a Pack containing a Destination:

1. Ensure the target Pack contains a valid Destination.
2. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet.
3. Navigate to **Routing**, then **Data Routes**.
4. Add a new Route or expand an existing Route.
5. In the **Destination** field, select your Pack from the menu.



Box labels distinguish Packs from other objects in the Destination field

6. Verify that data is flowing as expected.

A Pack only appears as an available target in the **Destination** field if it contains at least one configured Destination. System-provided Destinations, such as Default or DevNull, do not fulfill this requirement and will not make the Pack visible in the selection menu.

Send Data From a Pack to a Global Destination

To route data originating in a Pack to a Destination that exists outside a Pack at the Fleet level:

1. From inside the Pack context, select **Routes** from the top bar.
2. Add a new Route or expand an existing Route.
3. In the **Destination** field, select **Send to Worker Group Routes**. This sends any data matching the **Route** filter to the global Route table at the Fleet level.
4. Verify that data is flowing as expected.

Process Data in a Pipeline Contained in a Pack

To route data from a Source that exists at the Fleet level to a Pack containing a Pipeline:

1. Ensure the target Pack contains a valid Pipeline.
2. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet.
3. Navigate to **Routing**, then **Data Routes**.
4. Add a new Route or expand an existing Route.
5. In the **Pipeline** field, select your Pack from the menu.
6. Verify that data is processed as expected using the Data Preview window.

A Pack only appears as an available target in the **Pipeline** field if it contains a Pipeline.

16.1.3. Version Control Packs In External Git Repos

You can track and version-control Packs in an external Git repository, such as GitHub, GitLab, and Bitbucket. Tracking a Pack using an external Git repository establishes Git as the single source of truth for Pack configurations. This allows you to track configuration changes in a central, company-owned repository. It also makes it easier to integrate your Cribl deployment into your company's existing change management workflows.

Best Practices for External Pack Repositories

When tracking Packs in an external Git repository, keep these considerations in mind:

- **Each Pack needs its own Git repository.** To easily import and export Packs to an external Git repository, set up a separate Git repository for each Pack you want to version control.
- **Use branches for two-way synchronization:** While you can import a Pack from either a Git branch or a tag, tags are immutable and do not support pushing changes back to the remote repository. To enable a bidirectional workflow (where you both pull updates and push your local changes), you must use a branch.
- **Always pay attention to the Pack version.** Whenever you develop or export a Pack, get in the habit of checking the Pack version number in the Pack settings. When you make a change, check that you increased the version of the Pack before exporting. Cribl Edge only updates Packs when the version number has increased.
- **Consider protecting branches to prevent accidental deletions.** Set appropriate merge rights on these branches as needed.
- **Document what the Pack does:** If storing the Pack in an external Git repository, include README files and comments within components that explain what kind of data the Pack handles, which environment is it intended for, and whether there are any Sources/Destinations shared across Packs.

Export a Pack to an External Git Repository

After you have built your initial Pack, you can export it to an external Git repository and begin tracking it with Git version control tools. To get started, set up the connection between the Pack in your local environment and the version-controlled Pack in a Git repository.

Export a Pack

The first step is to export your Pack:

1. In Cribl Edge, navigate to the **Packs** page and open the Options ••• menu on the row of your Pack.
2. Select **Export** and choose the **Merge** export mode.
3. Confirm with **Export** and save the file locally.

Add the Pack to an External Git Repository

Now that you exported your Pack, you can add it to a new Git repository. You can use a public or private repository in any common Git hosting service, such as GitHub, GitLab, or Bitbucket. Regardless of whether your repository is public or private, you need to set up a personal access token for this repository, as explained in a later step: Set up a Personal Access Token and Import the Pack.

To add the Pack to an external repository:

1. Create an empty private repository in any hosting platform.
2. Optionally, create branches for your development, testing, and production branches. For example, `dev`, `test`, `prod`.
3. Clone the new repository to your local computer.
4. Extract the `.crbl` file into the root repository directory.

   Because they are gzipped, you can extract `.crbl` files using the `tar` utility. For example: `tar -xvzf WebdataStorage_0.0.1.crbl`.
5. Add, commit, and push the Pack content to your upstream Git repository.

Import and Update a Pack

In this step, you'll set up the connection between the Pack in your local Cribl Edge environment and the version-controlled Pack in a Git repository so that you can push and pull to branches from inside Cribl Edge.

You'll authenticate the connection by setting up a personal access token to connect Cribl Edge. Then, you'll make a change to the original Pack and push those changes to a development branch in your Git repository.

Set Up a Personal Access Token and Import the Pack

To import a Pack from Git into Cribl Edge, generate a personal access token for your Git hosting service:

1. Create a personal access token.
   Using GitHub as an example, create a personal access token with the `repo` scope.

   For GitHub only: You must authorize your personal access token to access the organization identity if you:

- Enabled SSO (single sign-on) authentication for your personal access token, and
- Are using a classic personal access token (as opposed to fine-grained).

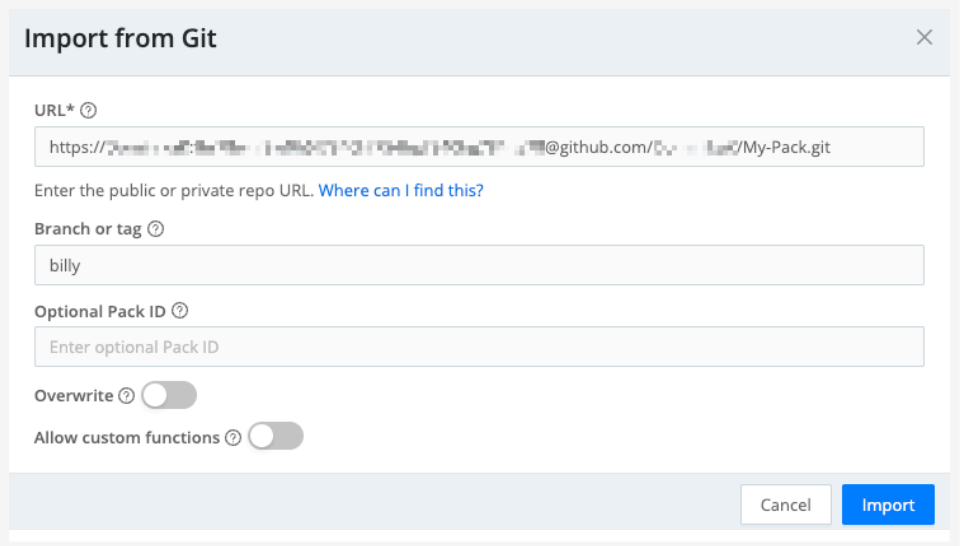
See the GitHub documentation about Authorizing a personal access token for use with single sign-on for specific instructions.

2. In Cribl Edge, select a Fleet and navigate to the **Packs** page.
3. Select **Add Pack** at the upper right and select **Import from Git**.
4. Enter the repository's URL. Provide your username and the token in this format: `https://<username>:<token>@github.com/<username>/<repo_name>.git`.
5. In **Branch or tag**, provide the name of your branch in the remote repository.

Choose your import source carefully. While you can import from a tag or a branch, only branches support pushing changes back to the remote repository.

- **Branches:** Use these for two-way synchronization (pull and push).
- **Tags:** Use these for read-only snapshots.

To switch from a tag to a branch, or to target a different branch for exports, you must delete the Pack and re-import it with the correct branch.



Importing a Pack from a Git repository

6. Select **Import** to confirm.
7. If the Pack you are importing binds variables to any fields, you will see a **Configure Variables** button in the notification about successful import. Select it to go to the **Variables** page, where you can configure the values for the variables for your deployment.
8. Check that the Pack is working correctly with your current environment based on its scope:

- If the scope of your Pack contains Sources and Destinations, verify data flow between everything included in the Pack.
- If your Pack only contains Routes and Pipelines, verify data flow by connecting the Pack to a Worker Group that has the necessary Sources and Destinations.

Verify data flow between Sources and Destinations both inside and outside the Pack.

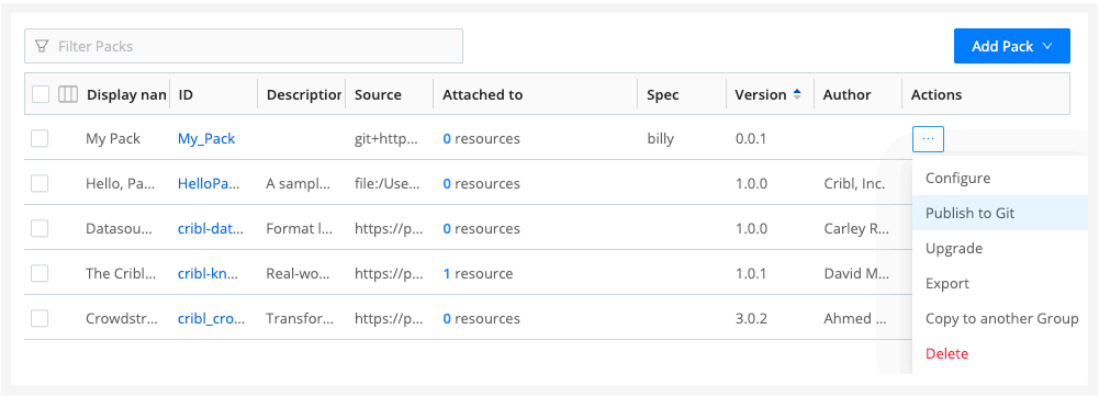


For on-prem deployments, it is possible to authenticate using SSH rather than personal access tokens. See Authenticate With SSH for more information.

Modify the Pack and Push Changes to Git

Now you can customize the Pack. Modify the Pack in some way by editing the Pack's configuration objects. For example, try making an edit to a Source, Destination, Route, Pipeline, Event Breaker, or Knowledge Object. For your first time using an external Git repository, keep the changes small. Then, publish your changes to the remote Git repository:

1. Navigate to the **Packs** page and open the Options ••• menu on the row of your Pack.



Publishing a Pack to Git

2. Select **Publish to Git**, and in the modal window, confirm with **Publish to Git** again.
3. Your Pack's branch in the remote repository updates with your changes.

The repository maintainer can now synchronize the main branch with your branch in their preferred way (for example, by creating a pull request).



When you use **Publish to Git**, all files in the repository are overwritten except for hidden files and directories located at the repository root. This means that any non-hidden files not managed by Cribl (such as documentation, scripts, or CI/CD workflows) will be deleted.

To preserve important files, ensure they are stored as hidden files by prefixing their names with a `.` For example use `.github` for GitHub workflows.

Authenticate With SSH (On-Prem Only)

For on-prem deployments, you can connect Packs to external Git repositories using SSH instead of HTTPS. This approach allows you to avoid embedding personal access tokens in your import URLs.

To authenticate with SSH:

1. Add your private SSH key to the Leader Node. See Connect to a Remote with SSH for more information.

   This SSH key must also have the necessary permissions (Read/Write) on your Git host provider (GitHub, GitLab, Bitbucket).

2. Select **Add Pack** at the upper right and select **Import from Git**.
3. Enter the repository's URL.
   When using SSH, the URL format differs slightly from the standard SSH Clone string provided by Git UI buttons. Specifically, you must use a `/` (slash) instead of a `:` (colon) after the domain. For example:

Authentication Method	URL Syntax
Standard HTTPS URL (with personal access token)	`https://<username>:<token>@github.com/<org>/<repo>.git`
Optimized SSH URL	`ssh://git@github.com/<org>/<repo>.git`

4. In **Branch or tag**, provide the name of your branch in the remote repository.
5. Select **Import** to confirm.

The alternative SSH authentication is not supported in Docker-based sandbox or ephemeral environments. For containerized deployments, continue authenticating with a personal access token.

Goat-to Resources

Time to GIT Serious - Automate Your Cribl Pack Deployments — Blog

16.2. Packs Publication Standards

This page outlines the process for Cribl Community members to publish Cribl Stream, Edge, and Search Packs to the Cribl Packs Dispensary. It also lists standards that apply to all publicly available Community Packs.

Publication Overview

Publishing your Pack is a five-stage process:

1. **Prepare**
    - Prior to starting development, read this topic's Pack Review Summary and Pack Review Checklist. These resources will help you create a Pack that can easily pass the review process. Decide on the problems you're trying to solve, and determine the dataset(s), data senders, and data receivers you're going to work with.
2. **Develop your Pack**
    - You'll generally develop Packs in your own Cribl environment, using saved sample data files to build out Stream/Edge Pipelines, or using Search Datasets to build Search Dashboards. During this stage, you might consider working collaboratively within Cribl's Community Slack `#packs` channel. When the Pack is ready for review, export it as a `.crbl` file, using the Cribl UI.
3. **Test your Pack**
    - Cribl strongly recommends that you test the Pack yourself, in your own environment. To verify that all content was exported correctly, import your Pack into a fresh Cribl environment. For Stream or Edge, this might be a newly created Docker instance, Worker Group, or Fleet. For Search, this might be a different Cribl Search Workspace. Upload the Pack, then use the Pack Review Checklist to verify that the Pack will pass review.
4. **Submit your Pack for review**
    - Sign into, or create an account on, the Cribl Packs Dispensary site. You can use your Cribl.Cloud account's email address to create the Cribl Packs Dispensary account.
    - If you don't have a Cribl.Cloud account, Cribl automatically creates a new one for you when you create your Cribl Packs Dispensary account.
    - Once you sign in, you'll see the Dispensary page with the **Publish Pack** button.



- Click **Publish Pack**, and then click **Upload Pack**.
- Select the `.crbl` Pack file you want to publish, and click **Submit**. You will be prompted to agree to the terms of Cribl's Pack Developer Agreement. As a Pack author, you must electronically acknowledge that you have read the Agreement, and that you intend to adhere to its requirements.

> Once you've uploaded the Pack, Cribl will review it. The review process typically takes a week or more. Once your Pack is reviewed, you'll receive an email from `packs@cribl.io` either informing you that it was accepted, or providing the reason(s) why it was rejected. (In the latter case, you can address the specified issues and resubmit.)

5. **Once your accepted Pack is published to the Cribl Packs Dispensary, it's time to celebrate!**

Pack Review Summary

To ensure consistency across published Packs, Cribl evaluates all submitted Packs against the Pack Review Checklist. Any Pack that fails to satisfy the checklist's requirements will be rejected. (As noted above, we will notify you of the basis for any rejection, so that you can remedy the issue and resubmit.)

Cribl recommends:

- Check your Pack against the checklist **before** trying to submit the Pack. This way, you can avoid having your Pack rejected for something easily fixed upfront.
- For Stream/Edge Packs, you can install the Syslog Pre-processing Pack as a reference - it includes clear examples of each item on the checklist.
- For Search Packs, you can install the Search Palo Alto Networks Traffic Pack as a reference.

Pack Review Checklist

This is a checklist for Packs you intend to submit to the Cribl Packs Dispensary. The following sections list setup requirements for product-specific Packs, and then for all Packs.

Stream/Edge Packs - Requirements

This section's requirements apply to Packs for Cribl Stream and Cribl Edge.

Sources and Destinations

If you include a Source (or Stream Collector) in your Pack, you must also include the intended Destination, so that the Routes table is wired up to receive and deliver data all within that Pack. If you include Sources (including Collectors) or Destinations inside a Pack, you must clearly state this in the README, and your README should warn users not to insert this Pack (as a Pipeline) into the global Routes table outside of the Pack.

The benefit of tying everything together in a Pack - Sources, Pipelines, Routes, Destinations - is enabling users to take advantage of an end-to-end telemetry stream configuration. This way, users need to update only selected values (for things like endpoints and authentication details) in order to run the Pack.

Routes

A submitted Stream or Edge Pack must include at least one Route in addition to the default Route.

- A Pack's Routes page must have a Route, with an appropriate filter, routing to a Pipeline.
- The filter should match **only** the set of data that the Pipeline is designed to process. The filter must **not** be set to `true`.
- If the Pack deals with multiple data types, each data type should have its own Route/Pipeline pair. (For a good example, look at the Palo Alto Networks Traffic Pack.)
- The Pack must not use the `default` Route to send events to a Pack-specific custom Pipeline. The `default` Route may have a Pipeline set to `main`, `devnull`, or `passthru` to handle events that do not match the Pack filter(s).
- Each Route must include a short description of the data sample being matched.
- Pairs of Routes and Pipelines should have matching names.

Pipelines

A Pack submitted must include at least one Pipeline in addition to the default Pipelines (which are `apache_sample_pipeline`, `devnull`, `main`, `passthru`, and `prometheus_metrics`.).

- If the Pack deals with multiple data types, there must be one Pipeline (and one Route) for each data type. (For a good example, look at the Palo Alto Networks Traffic Pack.)
- The Pack must not modify the default Pipelines. Leave these Pipelines unchanged.
- Every Pipeline must begin with a Comment Function that gives an overview of the Pipeline's operation as a whole. This Comment is a synopsis of what the entire Pipeline will be doing. For an example of the expected level of detail, refer to the Syslog Pre-processing Pack.
- If Function Groups are used within the Pipeline, each Function Group must begin with a Comment Function that gives an overview of what will happen within that Function Group.
- Every Function must have a Description that briefly explains exactly what is happening in that particular Function. During review, change the Pipeline view option to display the Description for each Function.

- Using the sample files provided with the Pack, validate that the Functions work as detailed in the Comment at the top of the Pipeline.

Sample Data Files

The Pack must include at least one sample data file for each of its Pipelines.

Sample data files:

- Must **not** have an expiration date.
- Must **never** include PII, customer data, or customer hostnames.
- Must be named to make it obvious which sample file(s) apply to each Pipeline.
- Must contain samples that are adequate for a customer to use to review the operation of a Pipeline. To allow performance analysis of the corresponding Pipeline, a sample should include 20 or more events, with 100 or more recommended.
- May be reused across Pipelines, but it is preferable to include multiple sample files, each specific to one Pipeline's Route and filter.
- Must work as expected on a clean installation of the Pack. This applies to all sample data files included in the Pack - Pack developers should test this before submitting the Pack.

Search Packs - Requirements

This section's requirements apply to Packs for Cribl Search.

Lookups

Ensure that lookup files contain no sensitive data. If Cribl review discovers any sensitive information in a submitted lookup file, Cribl will automatically reject the Pack.

When formatting references to the lookup table file, use this syntax:

```
pack(<pack_id>).<lookup_file_name>
```

Copy

When referencing the lookup table in queries, use this syntax:

```
dataset="cribl_search_sample" datatype="aws_vpcflow"
 | lookup pack(cribl-search-aws-vpc-flow-logs).protocol_codes on protocol=protocol_cc
 | summarize count=count() by protocol_name
 | order by count desc
```

Copy

Datatypes

Datatypes are important for Cribl Search Packs, because they provide the rules for how to process and search raw data. Adding Datatypes to Search Packs provides users with important parsing rules that enhance the overall search and Dashboard experience.

Essentially, Datatypes give structure to unstructured data, making it easier to query and explore vendor-specific datasets. Cribl strongly recommends that you implement Datatypes within Search Packs when needed. As a model, refer to the **Palo Alto Datatypes** ruleset that's built into Cribl Search.

All Products - Requirements

This section's naming and metadata requirements apply to all Packs.

Naming, Logo, and Tags

Every Pack must have both a unique **Pack ID** and a **Display name**.

When you view Packs in the Dispensary, the Display name appears above the Pack ID. The Pack ID must follow a specific format of sequenced tokens:

`<source>-<productArea>-<dataVendor>-<dataType>`.

Here:

- `<source>` identifies who contributed the Pack, not a data source. This is either `cribl` for Cribl-developed Packs, or `cc` for Community-contributed Packs. (Packs not satisfying this requirement will be immediately rejected.)
- `<productArea>` is one of `stream`, `edge`, or `search`.
- `<dataVendor>` and `<dataType>` identify (where applicable) the integration; and (where applicable) the focus of the transformations or configurations provided in the Pack. Examples for this pair of tokens might be `okta-rest`, or `palo-alto-networks-traffic`, or `exabeam-fortigate-firewall`.

Here are examples of complete Pack IDs from the Cribl Packs Dispensary (where you can find many additional examples):

- `cribl-stream-sumologic`
- `cribl-search-cloudtrail-activity`

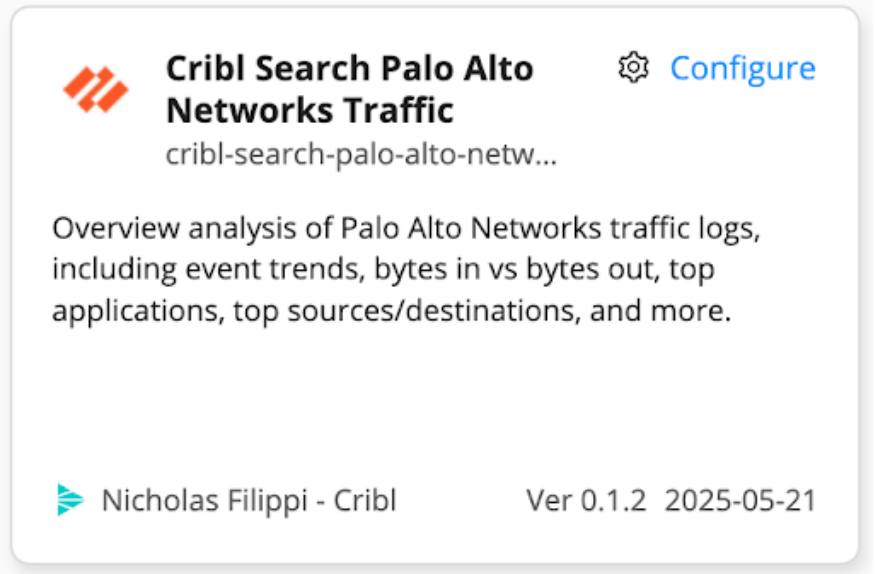
The Display name is a free-text summary of the purpose of the Pack. As examples:

- `Cribl Stream for Sumo Logic` is a Display name, while `cribl-stream-sumologic` is the corresponding Pack ID.
- `Cribl Search CloudTrail Activity` is a Display name, while `cribl-search-cloudtrail-activity` is the corresponding Pack ID:

Detailed requirements for both fields follow these graphical examples from Dispensary tiles.



Display Name and Pack ID - Stream or Edge example



Display Name and Pack ID - Search example

Pack ID Requirements

The Pack ID:

- Must start with either `cc-` (Community-contributed) or `cribl-` (Cribl-contributed). Packs not satisfying this requirement will be immediately rejected.
- Should (by convention) be all lowercase, using hyphens (`-`) as delimiters.
- Should **not** include the word "pack."

Display Name Requirements

The Display name is a friendly, human-readable version of the Pack name that:

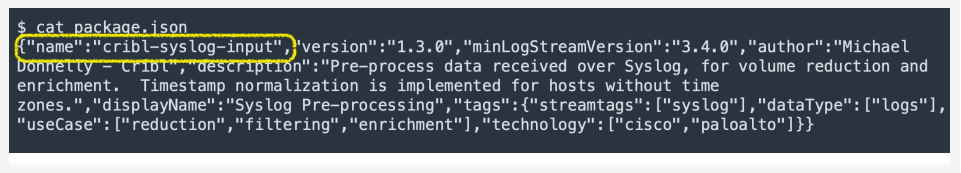
- Omits `cc-` and `cribl-`.
- Uses spaces where appropriate, and uses conventional capitalization.
- Matches the vendor's preferred spelling, spaces, and capitalization of any referenced company or product name.

Within a Pack, you can view or edit the Display name on the **Pack Settings** tab.

How to Review the Pack ID

This is the same procedure that the Cribl will perform when reviewing your Pack:

1. Extract the `.crbl` tarball that was submitted to the Dispensary:
   `tar zxvf <your_filename>.crbl`
2. View the `package.json` file:
   `cat package.json`
3. Check the value of the `name` field. This should be the Pack ID that you want.



Verifying the Pack ID

Pack Settings

You must review your **Pack Settings** to verify that the Pack satisfies all of the following criteria.

Display Name

A friendly, human-readable version of the Pack name, as defined above.

Version

The version number in the submitted Pack must have a corresponding entry in the README Release Notes. This number should be in SemVer (Semantic Versioning) format.

- The minimum version for initial publication is `0.1.0`. A Pack with a lower version **will be rejected**.
- A release that is considered "beta quality" should be designated as version `0.9.0`.

Description

Use this as a brief "hook" of one to two sentences. Example:

```
This pack for Syslog inputs will reduce volume, and address timestamp normalization for Syslog senders that omit timezones.
```

Author

Community-contributed Packs may include the author's name, a company name, or both.

All three of the following examples are valid:

1. `Jon Smith`
2. `Bazookatron`
3. `Jon Smith - Bazookatron`

Cribl-published Packs must use author's full name, plus `- Cribl`. Example: `Michael Donnelly - Cribl`.

Data Type, Use Cases, Technologies

Select all that apply.

Tags

Tags are optional but recommended, especially where multiple Packs are intended to work together. Tag examples include: `OCSF`, `AWS`.

Logo

The logo is optional, but recommended. You can include a logo associated with the technology addressed by the Pack. As examples: a Windows logo for Windows events, or an AWS Cloudwatch logo for AWS Cloudwatch data.

You can set the logo via **Pack Settings** > **Display**.

README

The `README` is the documentation others will use in selecting and deploying the Pack. The `README` requirements ensure that others will be able to successfully deploy the Pack.



The default sections for the README, as created when you click **Create Pack**, do not match the required and optional sections in the checklist. You will need to delete or replace certain sections.

On the Pack's **Pack Settings** > **README** tab, validate that the README text is correct. The following rules apply to all sections of the README:

- Proper grammar will help the users of your Pack. Use a grammar and spelling checker to make sure the README is clear and readable.
- If the README includes any text that looks like a placeholder, the Pack will be rejected.
- Trademark or product names must be spelled and capitalized exactly as the company that owns them would dictate. This requirement also applies to Pack settings, inline comments, and other uses. Examples: Amazon (not amazon), GCP (not Gcp), CrowdStrike (not Crowdstrike).

The README **must** include all of the sections enumerated here, as items **a** through **f**. The README may also include additional sections, as necessary to explain the resources you've included in the Pack.

a. About This Pack

Think of this initial section as the first paragraph of an essay, or as the only paragraph a person will read when deciding to use the Pack. It must clearly state what the Pack does. Start with the benefits of using the Pack. In sum, this section should answer the question, **Why should anyone use this Pack?**

If the Pack is tied to a specific data source, dataset, or destination, spell that out clearly here - even if that source or destination is in the Pack name.

b. Deployment

This **required** section must include all instructions necessary to get the Pack up and running. If the Pack requires any configuration after initial installation - but before it's put it into production - you must specify the configuration steps here.

Stream/Edge Pack Deployment

Filters: If applicable, say what filter must be used to route data to the Pack, or explain how to configure the filter. A filter based on `__inputId` is much more efficient than a match against `_raw`.

Sources: If the Pack requires a specific Source (or Stream Collector), include information on how to configure that Source. At a minimum, this needs to cover setting up the Source within Cribl. In some cases, this might include information on how to set up an external environment for collection by Cribl as well.

Destinations: If the Pack requires a specific Destination, include information on how to configure that Destination. Does it need a post-processing Pipeline tied to the Pack?

Pipelines: If the Pack includes Functions (within a Pipeline) that need to tailored, enabled, or disabled, include a full explanation and instructions.

Lookups: If the Pack includes Lookup files that need to be tailored, you must specify the configuration steps here.

Search Pack Deployment

Dataset Provider: If the Pack works with a Cribl Search built-in Dataset Provider, specify it. If the Pack requires adding a new Dataset Provider, provide specific instructions.

Dataset: Provide instructions on configuring compatible Datasets.

Datatype: Specify Datatypes to apply - whether built into Cribl Search or included in the Pack.

Macros: Provide instructions on using any custom Macros included in the Pack.

c. Upgrades

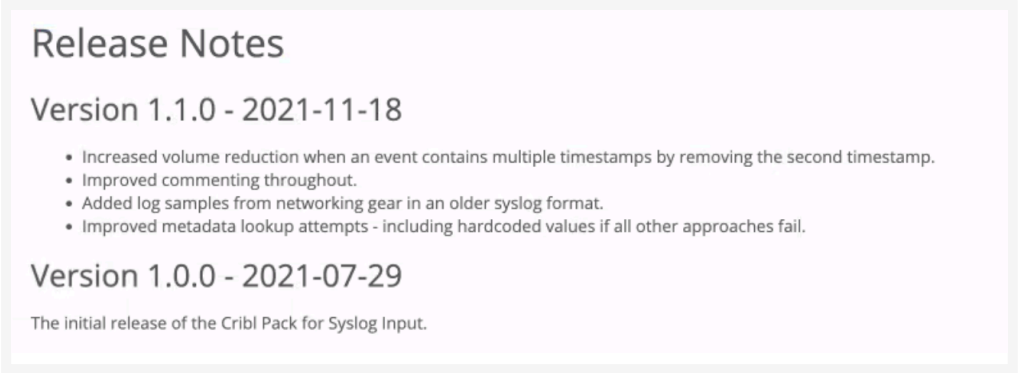
This **optional** section is strongly recommended if you expect others to customize their copy of the Pack and later upgrade the modified Pack.

- If included, this section should cover the usual set of problems with Pack upgrades as described in Upgrading an Existing Pack, and should explain the specifics of upgrading this particular Pack.

d. Release Notes

This **required** section is a list of release notes where each release note:

- Starts with the version number and date of release, separated by a space, a hyphen, and a space. For example: `Version 1.2.0 - 2022-07-11`.
- Continues with bullet points that describe what has changed and/or is new in the release.



Release notes

Release notes must adhere to the following rules:

- Designate the initial release version `Initial release.`
- The most current version listed must exactly match the Pack version in **Pack Settings** > **Pack Info**.
- Release notes versions must be listed starting with the most recent (in descending chronological order).

e. Contributing to the Pack

This **required** section explains how to contribute to the Pack.

- The information provided must be **actionable**. A section reading `[Contributor instructions go here.]` would be rejected.
- Cribl recommends that you refer potential contributors to Cribl's Community Slack. For example, Cribl-created Packs read as follows:

```
To contribute to the Pack, please connect with us on [Cribl Community Slack](htt
```

Copy

f. License

This **required** section must include text that designates **and links to** a legitimate license.

- Community-contributed Packs must include a license that allows for use by the general public, such as Apache 2.0, MIT, or GNU.
- Built-by-Cribl Packs must use the following snippet:

17. Reference

A reference for Cribl Edge, including:

- Cribl Expressions
- CLI
- Environment variables
- Config files

For reference documentation for the Cribl API, see the API Reference.

17.1. Knowledge Objects

Knowledge objects are reusable, customizable objects that help you scale and maintain your Cribl Edge more easily. You can reference these objects in multiple places, such as Functions or Pipelines. With these objects, you can change a single value in a central location and then populate that change everywhere you use that object. Access these objects in **Processing** > **Knowledge**.

Knowledge Object	Description
Lookups	Reference tables that you can apply to raw data events in your Pipeline for routing and enrichment.
Event Breakers	Event Breaker Rulesets are ordered collections of event-breaking rules that help you define the boundaries and structure of raw log data. Once you have converted unstructured raw data into structured events, you can then send them through Pipelines for further routing and data processing.
Parsers	Parsers for use in the Parser Function.
Global Variables	Store values you can reference elsewhere in Cribl Edge.
Sample Files	A data sample (often called a data capture) is a small, representative subset of live or stored data from a given data stream. When you creating samples as Knowledge objects, these files can be accessed by Routes, Pipelines, Event Breakers, and Datagens at the Fleet or Pack level.
Regexes	Regular expressions for use in regex-related Functions.
Grok Patterns	Grok pattern files for use in the Grok Function.
Schemas (JSON)	JSON schemas for validating events.
Parquet Schemas	Schemas for writing out data in the Apache Parquet format.
Database Connections	(**Cribl Stream only**) Reusable configuration objects that enable Database Collectors to access and retrieve data from a MySQL, Oracle, Postgres, or SQL Server database.
HMAC Functions	(**Cribl Stream only**) Keyed hashing functions that you use with REST Collectors.

17.1.1. Variables Library

Variables store values you can reference elsewhere in Cribl Edge. These variables allow you to change a single value and then populate everywhere that variable is referenced.

Variables are an important component for scaling your Cribl deployment and improving ease of maintenance. Typical use cases for variables include:

- Storing a constant that you can reference from any Function in any Pipeline.
- Storing a relatively long value expression, or one that uses one or more arguments.
- Converting abstract, complex logical expressions into a reusable variable that has a human-readable name for easier maintenance.

Cribl Edge supports a variety of variable types:

- **Number, String, Boolean:** Store simple, static configuration values for reuse and ease of updating (such as threshold limits or names).
- **Encrypted String:** Store sensitive strings (like API keys or passwords). The value is encrypted at rest but retrieved as plain text when referenced, including when you are logged into the Cribl interface.
- **Object:** Store complex, structured data or key-value dictionaries, often used as an alternative to small lookup files.
- **Array:** Store a list of items (such as IP ranges or required field names).
- **Expression:** Define complex, repeatable logic that requires calculated output, especially when the logic needs to accept arguments.

You access all variables using the standard syntax `C.vars.<variableName>`. See Cribl Expressions for more information about `C.vars` and other Cribl Expressions.



This page explains the general use case for variables. For more information about using variables in Packs, see:

- Pack Variables
- Configure Variables and Secrets

Limitations

When multiple variables share the same name, Cribl Edge resolves the value based on scope, moving from the most specific (inner) scope to the most general (outer) scope. In other words: local scope values win over

broader scopes.

For example, a variable defined inside a Pack (which has a local scope) will always override a variable with the same name defined in the Fleet settings.

Create a Variable

Creating a new variable has three main phases:

1. Define the variable scope.
2. Configure the variable.
3. Reference the variable in a Function or Pipeline.

Define the Variable Scope

Variables are scoped to different levels of the application. You can only use a variable in the scope in which it is defined.

When creating a new variable, you determine its scope based on where you access and create the variable:

Fleet | Pack | Single-Instance Deployment

To access and create variables specific to a given Fleet:

1. Select **Fleets** from the sidebar, and choose a Fleet.
2. On the **Fleets** submenu, select **More**, then **Knowledge**.
3. In the sidebar, select **Variables**.



You cannot access variables from a Pack in a Fleet.

Configure the Variable

After ensuring you have accessed the variable from the correct scope, create a new variable:

1. Select **Add Variable**.
2. Configure the variable settings:
    - **Name:** Enter a unique name for the variable.
    - **Description**: Optionally, enter a description.

- **Type:** Select the type of variable you want to create. Options include:

Type	Explanation	Use Cases
String	A sequence of characters (text).	Can centralize environment identifiers, file paths, URL prefixes, or standard naming conventions. If a string or name changes depending on its context (such as a different environment), you only have to update it in the variable and it populates to configurations in that environment.
Number	For numeric values, including integers (whole numbers) and floating-point numbers (decimals).	Ideal for defining thresholds or limits that might need tuning later.
Encrypted string	Stores a string value that is automatically encrypted when saved to the configuration file on disk (YAML). However, within the application at runtime, it is decrypted and treated as a standard string.	Use for storing sensitive credentials like API tokens, secret keys, or passwords. This ensures that if someone inspects your configuration files or git commits, they only see the encrypted hash, not the plaintext secret.
Boolean	Represents a binary logical value that resolves to `true` or `false`.	Use as a switch to enable or disable specific logic paths, detailed logging, or experimental features across your entire deployment without editing individual Pipelines.
Array	An ordered list of values (strings, numbers) stored in a single variable.	Can help maintain long lists of items, such as IP ranges, required field names, known good or bad hostnames, and more.
Object	A structured collection of key-value pairs (dictionaries).	Functions as a lightweight, in-memory lookup table.
Expression	A snippet of JavaScript code that is evaluated only when called. Expressions can accept arguments, allowing them to act like reusable custom logical functions.	Allows you to encapsulate complex, multi-condition logic (like regex matching) into a single function with a human-readable name.
Any	A flexible container that accepts a value of any variable data type (string, number, expression) without enforcing strict validation rules.	While this variable type is flexible, it removes the safety net of UI validation. Use specific types (like number or boolean) whenever possible for better reliability.

- **Value:** Enter the value, data structure, or JavaScript expression for this variable. This input must match the format required by the selected variable type. For example, Expressions should have JavaScript syntax. The system validates the input when you save.
- **Tags:** Optionally, add tags that you can use for filtering and grouping in the variable list view. Use a tab or hard return between (arbitrary) tag names.

3. Save the variable. The system validates the **Value** field to ensure it matches the specified **Type** when it saves.

Reference the Variable

After creating the variable, you can start calling it in Functions, Pipelines, and Packs. You can reference a variable in fields that let you use custom JavaScript code. See the introduction of Build Custom Logic to Route and Process Your Data for examples and images of JavaScript-enabled fields.

Some variable types require additional inputs or arguments:

Variable Type	How to Reference	Examples
Number, String, Boolean	Use `C.vars.variable_name` in any field that lets you use custom JavaScript code.	`C.vars.my_variable`
Encrypted String	Use `C.vars.variable_name` in any field that lets you use custom JavaScript code.	`C.vars.api_key`
Object	Reference these as you would any other object or array using standard JavaScript object notation.	`C.vars.lookup_map.key` `C.vars.my_config.region`

Array	Reference these as you would any other object or array using standard JavaScript object notation.	`C.vars.varname[0]`
		`C.vars.varname.nested`
		`C.vars.host_list[0]`

Expression	Call it with the arguments in parentheses.	`C.vars.varname(arg1,arg2)`

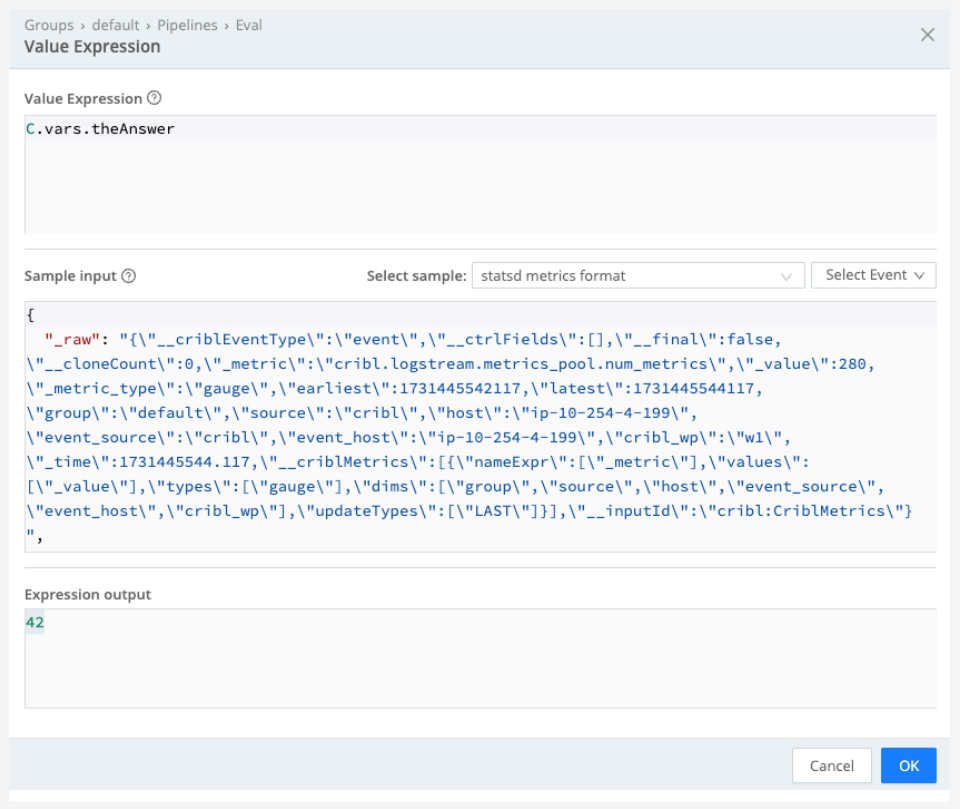
To illustrate the workflow for referencing a variable, imagine a scenario in which you want to add a field named `myField` to all incoming events. This field value will get referenced in multiple Pipelines and may possibly change, which is why you decided to make it a variable named `theAnswer`. For now, you want to set it to a static number value of `42`.

After creating the number variable named `theAnswer` with a value of `42`, you could then reference it in a Pipeline:

1. Add the Eval Function. (The Eval Function adds, modifies, or removes event fields.)
2. Select **Add Field** to add a new field.
3. Configure the new field to reference the `theAnswer` variable:

- **Name:** myField
- **Value Expression:** `C.vars.theAnswer`

4. To validate the value, select **Advanced Mode** in the **Value Expression**.
5. Confirm that the **Expression output** is `42` as expected.



Value Expression Advanced Mode

You can then reference this variable in multiple Functions or Pipelines as needed. When you change the variable value, it will update anywhere the variable is referenced.

> Fleet variables are designed specifically for configuration binding. They are not accessible using the C.vars expression in Pipeline Functions. You can only bind these variables to Source and Destination configuration fields that display the **Variable** icon .
>
> The only exception is the Redis Function, which can resolve Fleet variables at runtime to manage external state or lookups.

Use Case Examples

You can personally test these examples within your own Cribl Edge deployment to understand how Cribl evaluates variables.

Expression Example

This example shows an expression variable that executes JavaScript logic to get the current time and assign it to a field named `nowEpoch`. This expression does not require any input arguments, so it gets left blank.

Variable settings:

- **Name:** `epochTime`
- **Type:** Expression
- **Value:** `Date.now()/1000`

You could then add an Eval Function to a Pipeline:

- **Field Name:** nowEpoch
- **Value Expression:** `C.vars.epochTime()`

Object Example

This example demonstrates how to use an Object variable as a central configuration store (dictionary). You can reference specific keys within the object using dot notation. In this scenario, the goal is to assign the field `contact_email` a value that is stored in the `admin_email` key of a global configuration object.

Variable settings:

- **Name:** `my_app_config`
- **Type:** Object
- **Value:**

```
{
  "env": "production",
  "retry_count": 3,
  "admin_email": "ops@example.com"
}
```

Copy

You could then add an Eval Function to a Pipeline:

- **Field Name:** contact_email
- **Value Expression:** `C.vars.my_app_config.admin_email`

Array Example

This example shows how to use an Array variable to store a list of values (like hostnames) and check if a specific event field exists within that list. In this scenario, the goal is to drop the event if the host field matches any entry in the `blocked_hosts` list. This will be accomplished with logic that returns `false` in a filter.

Variable settings:

- **Name:** `blocked_hosts`
- **Type:** Array
- **Value:** `["bad-host-1", "test-server-99", "retired-asset"]`

You could then add a Drop Function to a Pipeline with a **Filter** value set to: `C.vars.blocked_hosts.includes(host)`.

Additional Examples

There are endless possibilities for how to use variables in Cribl Edge. See more examples in the Goat-to Resources and continue sharing use cases with the rest of the Cribl Community on Slack.

Goat-to Resources

Using Global Variables to Simplify, Automate, and Save — Blog

17.1.2. Regexes

Regexes offer predefined regular expressions that you can use in regex-related Functions.

Cribl ships with a collection of commonly-used regexes and gives you the possibility of adding your own custom ones.

Regexes let you quickly identify and filter data that conforms to specific patterns, including commonly-used patterns like credit card numbers, IP addresses, or personal identification numbers. You can use expressions from the regex collection in several Functions, such as Regex Filter, Regex Extract, or Mask, in configuration for certain Sources and Destinations, as well as in other settings.

Prerequisites

You configure regexes per Fleet, or within a Pack. You can only use the regex within its scope, that is, in the Fleet or Pack you defined it in.

Regexes defined in an Edge Fleet are subject to Fleet inheritance.

Regex Priority

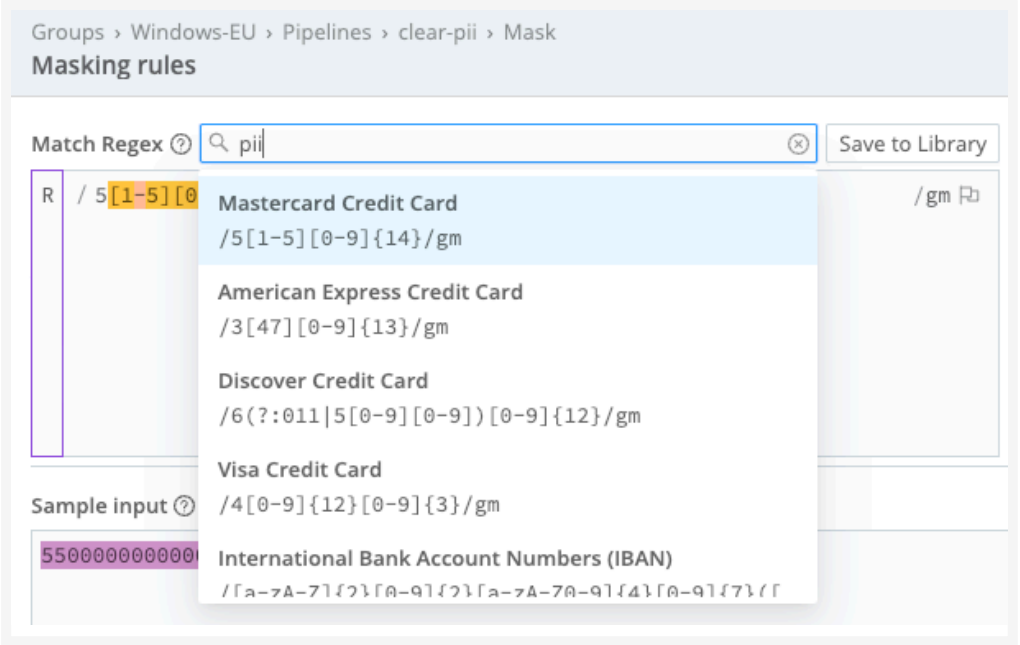
Built-in Regexes are listed with the **Cribl** tag, while those you add are tagged with **Custom**. When you edit any built-in regex, it changes to a **Custom** one as well.

If two regexes with the same ID exist, the custom pattern takes priority in listings and search. For example, if a Cribl-provided pattern and a Custom one are both named `ipv4`, the one from Cribl will not be displayed or delivered as a search result.

Use a Regex

To insert a regex into a field that supports it, for example in a Function, select the Advanced mode or Edit icon beside that field.

In the resulting modal, regexes will appear as typeahead options. Select a pattern to paste it in. You can then use the pattern as-is, or modify it as necessary.



Using a built-in regex

Create a Regex

To create a new regex:

1. In the sidebar, select **Fleets** and choose a Fleet.
2. In the **Fleets** submenu, select **More**.
3. If you are adding regex to a Pack, select **Packs** and choose your Pack.
4. Select **Knowledge**, then **Regexes**. You will see a list of prebuilt and custom regexes.
5. Select **Add Regex**.
6. Enter an **ID** for the regex and an optional **Description**.
7. In **Regex pattern**, enter the pattern, using the JavaScript (ECMAScript) regex flavor. Use the flag icon to select options for regex matching.
8. In **Sample data**, you can enter a sample text to check the expression against. Matches will be indicated visually.



Adding a custom regex

9. Optionally, add one or more **Tags** that will help you search for this regex.
10. Select **Save**.

Examples

Mask SSN Numbers

If you need to mask personal identification numbers, you can use a build-in regex in conjunction with the Mask Function.

Add a Mask Function to your Pipeline. In the Function's **Masking Rules** modal, search and select the **US Social Security Numbers** regex in the **Match Regex** box. In **Replace Expression**, select the string or expression you want to mask the number with.

Mask Other Personal Identity Numbers

17.1.3. JSON Schemas

Cribl Edge uses JSON schemas for validating JSON events.

You can use the built-in Cribl Schema or create custom ones and use them to validate any fields by using the `C.Schema` expression.

JSON schemas are based on the popular JSON Schema standard, and Cribl Edge supports schemas matching that standard's Drafts 0 through 7.

You can refer to a single JSON schema in multiple expressions. When editing them, be aware that any change will affect all places it is used in.



Cribl Edge supports two kinds of schemas:

- JSON schemas for validating JSON events, described on this page.
- Parquet schemas for writing data from a Cribl Edge Destination to Parquet files.

These schemas serve different purposes. Operations that work for one kind of schema can't be used with the other. For example, the validation method for JSON schemas can't be used to validate Parquet schemas.

Validate an Object Against the Schema

To validate an object against a schema, use the `C.Schema` expression: `C.Schema('<schema_name>').validate(<object_field>)`.

You can call this method anywhere in Cribl Edge that supports JavaScript expressions. Typical use cases for schema validation include:

- Making a decision before sending an event down to a Destination.
- Making a decision before accepting an event (for example, dropping an event if invalid).
- Making a decision to route an event based on the result of validation.

Add New Schema

To add a new JSON schema:

1. In the sidebar, select **Fleets** and choose a Fleet.

2. In the **Fleets** submenu, select **More**.
3. If you are adding the schema to a Pack, select **Packs** and choose your Pack.
4. Select **Knowledge**, then **Schemas**.
5. Select **Add Schema**.
6. Enter an **ID** for the regex and an optional **Description**.
7. In **Schema**, paste the JSON schema content.
8. Select **Save**.

JSON Schema Example

To add this example JSON Schema, enter the following into **New Schema** fields:

- ID: `schema1`.
- Description: (Enter your own description here.)
- Schema: Paste the following schema.

JSON Schema - Sample

```
{
  "$id": "https://example.com/person.schema.json",
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "Person",
  "type": "object",
  "required": ["firstName", "lastName", "age"],
  "properties": {
    "firstName": {
      "type": "string",
      "description": "The person's first name."
    },
    "lastName": {
      "type": "string",
      "description": "The person's last name."
    },
    "age": {
      "description": "Age in years which must be equal to or greater than zero.",
      "type": "integer",
      "minimum": 0,
      "maximum": 42
    }
  }
}
```

Copy

Assume that events look like this:

Events

```
{"employee":{"firstName": "John", "lastName": "Doe", "age": 21}}
{"employee":{"firstName": "John", "lastName": "Doe", "age": 43}}
{"employee":{"firstName": "John", "lastName": "Doe"}}
```

Copy

To validate whether the `employee` field is valid per `schema1`, we can use the following:

`C.Schema('schema1').validate(employee)`

17.1.4. Parquet Schemas

Selected Cribl Edge Destinations support writing out data as files in the Apache Parquet columnar storage format.

The following Destinations support the Parquet format:

- Azure Blob Storage
- Azure Data Explorer
- Cloudflare R2
- Databricks
- Filesystem/NFS
- Google Cloud Storage
- MinIO
- Amazon S3 Compatible Stores
- Amazon Security Lake

You can select the Parquet format from the Destination's **General** > **Data format** drop-down. (An exception is the Amazon Security Lake Destination, which does not have the drop-down, because it uses the Parquet format exclusively). When the `Parquet` format is selected, you can configure the Parquet settings for the Destination, including selecting the Parquet schema to use.

You can automatically generate a Parquet schema based on the events of each Parquet file the Destination writes, by toggling on **Parquet Settings** > **Automatic schema** for the Destination. The automatically-generated schema preserves the exact structure of the ingested events, with no data lost. Every field in the generated schema will have a data type of `JSON`, `STRING`, `INT_64`, `UNSIGNED_INT_64`, or `DOUBLE`.

You can refer to a single Parquet schema in multiple Destinations. When editing them, be aware that any change will affect all places it is used in.

When logging is set to `debug`, you will see what schema is associated with the Parquet files that Cribl Edge writes out. On Linux, you can use the Cribl Edge CLI's `parquet` command to view a Parquet file, its metadata, or its schema.



Cribl Edge supports two kinds of schemas:

- Parquet schemas for writing data from a Cribl Edge Destination to Parquet files, described on this page.
- JSON schemas for validating JSON events.

> These schemas serve different purposes. Operations that work for one kind of schema can't be used with the other. For example, the validation method for JSON schemas can't be used to validate Parquet schemas.

Automatic Parquet Schema Generation

When you decide whether or not to generate Parquet schemas automatically, consider these pros and cons.

Advantages:

- Configuration is minimal.
- There's no need to know how events you want to ingest are structured.
- There's no need to change Parquet schema when event structure changes.

Disadvantages:

- Writing Parquet files takes longer than with predefined schemas. Note that schema complexity increases linearly with the number of events used to generate the schema.
- Automatically-generated Parquet schemas cannot be as fine-tuned as predefined ones, to capture certain nuances. For example, encoding will always be `PLAIN`, compression will always be `SNAPPY`.
- Some aspects of event structure (such as maps and sets) cannot be captured in the schema.
- Some aspects of time-related fields cannot be captured in the schema.
- Because the system produces a new Parquet schema for each file written, there may be differences between successive schemas. This can limit your ability to concatenate a collection of Parquet files.

Add Parquet Schema

If you are not using automatic schema generation, or if there's no pre-existing schema that suits your data, before configuring a Destination for Parquet output, you should add an existing Parquet schema, or create a new Parquet schema that suits the data you're working with.

You do not need to start from scratch: Cribl provides sample Parquet schemas for you to clone and then customize as needed.

To add a new Parquet schema:

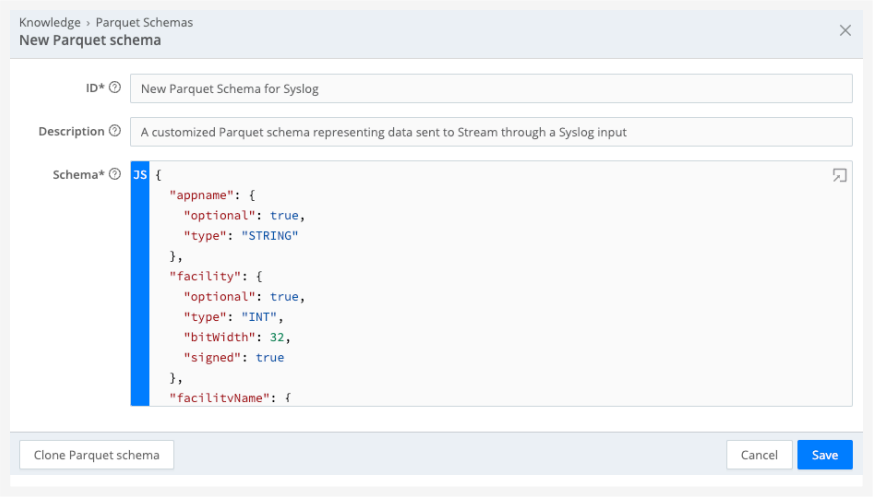
1. In the sidebar, select **Fleets** and choose a Fleet.
2. In the **Fleets** submenu, select **More**.
3. If you are adding the schema to a Pack, select **Packs** and choose your Pack.
4. Select **Knowledge**, then **Parquet Schemas**.

5. Select **Add Parquet schema**.
6. Enter an **ID** for the regex and an optional **Description**.
7. In **Schema**, paste the Parquet schema content.
8. Select **Save**.

Modify Parquet Schema

You can modify existing Parquet schemas, but Cribl strongly recommends that you first clone the schema, and give the clone its own distinct name.

1. Select the schema you want to start with, to open it in a modal.
2. Select **Clone Parquet Schema** to open the **New Parquet schema** modal.
3. Give the new schema a name and description.



Creating a Parquet schema

Cloning vs Modifying Schemas

Modifying an existing schema does **not** propagate your modifications to its clones. Cloning and renaming schemas is the safest approach, because it ensures that your Destination will use the newly modified version of the schema.

If you do not clone and rename the schema (that is, you leave the schema name unchanged), you still must re-select the schema in the Destination's **Parquet schema** drop-down to bring the modified version into the Destination.

Parquet Format Used by Cribl Edge

Different Parquet readers and writers behave differently. Keep the following guidelines in mind when working with Parquet in Cribl Edge.

In the Parquet Schema editor, you express your Parquet schema in JSON. This does not look like the examples in the Parquet spec, which have their own syntax (like the repetition/name/type triple), but they are functionally equivalent. See the examples below.

The editor provides autocompletion and validation to guide you. (However, Cribl Edge currently does not fully support autocompletion on deeply nested schemas.)

File Extensions

Cribl Edge can read Parquet files only if they have the extension `.parquet`, `.parq`, or `.pqt`.

Field Content

When Cribl Edge writes to a Parquet file:

- If the data contains a field that **is not** present in the schema - i.e, an extra field - Cribl Edge writes out the parent rows, but omits the extra field.
- If the data contains a field that **is** present in the Parquet schema, but whose properties violate the schema, Cribl Edge treats this as a data mismatch. Cribl Edge drops the rows containing that field - it does not write those rows to the output Parquet file at all.
- If the data contains JSON, the JSON must be stringified. Otherwise, Cribl Edge treats this as a data mismatch, and does not write out the row. For example, this valid (but not stringified) JSON will trigger a data mismatch: `{ "name": "test"}`.
  The same JSON in stringified form will work fine: `"{\"name\": \"test\"}"`.

Data Types

Cribl Edge supports:

- All primitive types.
- All logical types.
- All converted types.

Converted types have been superseded by logical types, as described in the Apache Parquet docs. Cribl Edge can read Parquet files that use converted types, but will write out the same data using corresponding logical types.

Repetition Type

You have three alternatives when defining a field's Repetition type:

- Set `optional` to `true`.
- Set `repeated` to `true`.
- Set neither `optional` nor `repeated`. This **implicitly** sets the Repetition type to `required`, and it is the default.

Usage guidelines:

- Do not set both `optional` **and** `repeated` to `true`.
- Do not use the `required` key at all.
- Instead of omitting `optional`, you have the option to include it, but set it to `false`.
- Instead of omitting `repeated`, you have the option to include it, but set it to `false`.
- If any field's Repetition type is `repeated`, Cribl Edge represents this field as a single key whose value is an array - not as separate key-value pairs with identical keys.

Encodings

- Among the `*DICTIONARY` encodings, Cribl Edge supports only `DICTIONARY`. Trying to assign the unsupported encodings `PLAIN_DICTIONARY` or `RLE_DICTIONARY` will produce an error.
- `BYTE_STREAM_SPLIT` encoding can be used only with `DOUBLE` or `FLOAT` types, and otherwise produces errors.
- The `RLE` and all `DELTA*` encodings also produce errors.

Parquet Limitations

Cribl Edge does not support writing Parquet files which employ any of the following:

- Parquet Modular Encryption.
- The Parquet Bloom Filter.
- Separation of metadata and/or column data into multiple files.

Cribl Edge does not support reading or writing Parquet files which employ the deprecated `INT96` data type.

Examples of Parquet Schema as JSON

These examples should give you a hint of how to express Parquet schema in JSON. For a full explanation of the Parquet syntax, see the Parquet spec.

List Example

In this example, the whole schema is named `the_list`, and its structure consists of a container named `list` whose zero or more elements are each named `element`. Note that in Parquet schema, to specify that a field is nested (that is, contains other fields), you just give it the type `group`.

Parquet Schema Example 1

```
message schema {
  REQUIRED group the_list (LIST) {
    REPEATED group list {
      REQUIRED BYTE_ARRAY element (STRING);
    }
  }
}
```

Copy

Expressed as JSON, the same schema is more spread-out, because to specify that a field is nested, you give it a sub-field named `fields`. (This is equivalent to declaring the nested field's type as `group`.)

Parquet Schema Example 1 Expressed as JSON

```
{
  "the_list": {
    "type": "LIST",
    "fields": {
      "list": {
        "repeated": true,
        "fields": {
          "element": {
            "type": "STRING"
          }
        }
      }
    }
  }
}
```

Copy

Map Example

In this example, the whole schema is named `the_map`, and its structure consists of a container named `key_value` whose zero or more pairs of elements each contain a `key` and a `value` element. Both the `key` and the `value` are strings in this case, but they could be other data types.

Parquet Schema Example 2

```
message schema {
  REQUIRED group the_map (MAP) {
    REPEATED group key_value {
      REQUIRED BYTE_ARRAY key (STRING);
      REQUIRED BYTE_ARRAY value (STRING);
    }
  }
}
```

Copy

Parquet Schema Example 2 Expressed as JSON

```
{
  "the_map": {
    "type": "MAP",
    "fields": {
      "key_value": {
        "repeated": true,
        "fields": {
          "key": {
            "type": "STRING"
          },
          "value": {
            "type": "STRING"
          }
        }
      }
    }
  }
}
```

Copy

Array of Arrays Example

In this example, the whole schema is named `array_of_arrays`, and its structure consists of a container named `list` whose zero or more elements - which are themselves lists - are each named `element`. Within each of those `element` lists are one or more additional `list/element` structures.

Parquet Schema Example 3

```
message schema {
  REQUIRED group array_of_arrays (LIST) {
    REPEATED group list {
      REQUIRED group element (LIST) {
        REPEATED group list {
          REQUIRED INT64 element;
        }
      }
    }
  }
}
```

Copy

Expressed as JSON, the same schema is cumbersome to read compared to Parquet, which was designed to express nested structures concisely.

Parquet Schema Example 3 Expressed as JSON

```
{
  "array_of_arrays": {
    "type": "LIST",
    "fields": {
      "list": {
        "repeated": true,
        "fields": {
          "element": {
            "type": "LIST",
```

17.1.5. AppScope Configs

AppScope is deprecated and will be removed in a future release. Don't use AppScope in production and ensure it is not part of your data flow configuration. For details, please read the announcement in Cribl Community Slack.

17.2. Cribl Expressions

Native Cribl Edge methods can be found under `C.*`, and can be invoked from any Function that allows for expression evaluations. For example, to create a field that is the SHA1 of a another field's value, you can use the Eval Function with this **Evaluate Fields** pair:

Name	Value Expression
myNewField	`C.Mask.sha1(myOtherField)`

> Where fields' names contain special characters, you can reference them using the `__e['<field-name-here>']` convention. For details, see Build Custom Logic to Route and Process Your Data: Fields with Non-Alphanumeric Characters.

Cribl expressions offer methods and properties in the following classes:

- C.Crypto - data encryption and decryption
- C.Encode and C.Decode - encoding and decoding data
- C.Lookup - running lookups
- C.Mask - masking sensitive data
- C.Net - network methods
- C.Text - text manipulation
- C.Time - time and date manipulation
- C.Schema - schema validation
- C.Secret - secret management
- C.env - environment properties
- C.os - system methods
- C.vars - global variables
- C.version - Cribl Edge versions
- C.Misc - various other methods

17.2.1. C.Crypto - Data Encryption And Decryption

C.Crypto.decrypt()

```
C.Crypto.decrypt(value: string, escape: boolean = false, escapeSeq = '"'): string
```

Copy

Decrypts all occurrences of ciphers in the given value. Instances that cannot be decrypted (for any reason) are left intact.

Returns `value` with ciphers decrypted.

Parameter	Type	Description
`value`	string	String in which to look for ciphers.
`escape`	boolean	Set to `true` to escape double quotes in output after decryption (for example, for data encrypted in Splunk.) Defaults to `false`.
`escapeSeq`	string	String used to escape double quotes. The default `'"'` escapes CSV output.

Examples

To decrypt the contents of the `encrypted` field, use:

```
C.Crypto.decrypt(encrypted)
```

Copy

C.Crypto.encrypt()

```
C.Crypto.encrypt(value: Buffer | string, keyclass: number, keyId?: string, defaultVal
```

Copy

Encrypts the given value with the `keyId`, or with a `keyId` picked up automatically based on `keyclass`.

Returns the encrypted value. If encryption does not succeed, returns `defaultVal` if specified; otherwise, `value`.

Parameter	Type	Description
`value`	string | Buffer	The value to encrypt.
`keyclass`	number	Key Class to pick a key from, if `keyId` isn't specified.
`keyId`	string	Encryption `keyId`, takes precedence over `keyclass`.
`defaultVal`	string	Value to return if encryption fails; if unspecified, the original value is returned.



You can decrypt the encrypted data in Cribl Search, using the `encrypt` and `decrypt` functions in your queries.

Examples

To encrypt the `customer` field using key with `keyId` 2, use:

```
C.Crypto.encrypt(customer, -1, 2)
```

Copy

To encrypt the same field with a key selected from the Key Class 3, without specifying a `keyId`:

```
C.Crypto.encrypt(customer, 3)
```

Copy

C.Crypto.createHmac()

```
C.Crypto.createHmac(value: string | Buffer, secret: string, algorithm: string = 'sha2
```

Copy

Generates an HMAC that can be added to events, or can be used to validate events that contain an HMAC.

Returns the calculated HMAC digest on success; otherwise, `value`.

Parameter	Type	Description
`value`	string | Buffer	The data to encrypt. When the `outputFormat` is invalid or undefined, this parameter is returned as the digest, via a Buffer.
`secret`	string	The secret key used to generate the MAC.
`algorithm`	string	The hash algorithm used to generate the MAC. Defaults to `'sha256'`. Run `openssl list -digest-algorithms` to see the list of available algorithms.
`outputFormat`	`'base64'` | `'hex'` | `'latin1'`	Output format for the MAC. Defaults to `'hex'`.

Examples

To generate HMAC from the `customer` field, provide the secret as the second argument:

```
C.Crypto.createHmac(customer, 'kETvYtLKZkgIzBvfRdVcZULBITANjOMd')
```

Copy

To additionally specify that you want to use the SHA512 hash algorithm and to output the HMAC in `latin1` format, run:

```
C.Crypto.createHmac(customer, 'kETvYtLKZkgIzBvfRdVcZULBITANjOMd', 'sha512', 'latin1')
```

Copy

17.2.2. C.Decode And C.Encode - Encoding And Decoding

C.Decode - Data Decoding Methods

C.Decode.base64()

```
C.Decode.base64(val: string, resultEnc: string = 'utf8'): string | Buffer | undefinec
```

Copy

Performs Base64 decoding of the given string.

Returns a string or Buffer, depending on the `resultEnc` value.

Parameter	Type	Description
`val`	string	Value to decode.
`resultEnc`	`'utf8' | 'utf8-valid' | 'buffer'`	Encoding to use to convert the binary data to a string. Use `'utf8-valid'` to validate that result is valid UTF8; use `'buffer'` if you need the binary data in a Buffer. Defaults to `'utf8'`.

Examples

To decode an `encoded` field and return the decoded value as a Buffer, use:

```
C.Decode.base64(encoded, 'buffer')
```

Copy

C.Decode.gzip()

```
C.Decode.gzip(value: Buffer | string, encoding?: BufferEncoding) : string
```

Copy

Gunzips the supplied value.

Parameter	Type	Description
`value`	any	Value to gunzip.
`encoding`	string	Encoding of `value`, for example: `'base64'`, `'hex'`, `'utf-8'`, `'binary'`. Default is `'base64'`. If data is received as Buffer (from gzip with encoding: `'none'`), decoding is skipped.

Example

To decode a `hostname` field which had been encoded using `hex`, run:

```
C.Decode.gzip(hostname, 'hex')
```

Copy

C.Decode.hex()

```
C.Decode.hex(val: string): number
```

Copy

Performs hex to number conversion. Returns `NaN` if value cannot be converted to a number.

Parameter	Type	Description
`val`	string	Hex string to parse to a number (for example, `0xcafe`).

C.Decode.inflate()

```
C.Decode.inflate(value: Buffer | string, encoding: BufferEncoding = 'base64', isRaw =
```

Copy

Inflates the supplied value.

Parameter	Type	Description
`value`	string | Buffer	The value to inflate. May be a Buffer or a string.
`encoding`	string	If a string `value` is provided, this specifies how it has been encoded. If no `encoding` is provided, `base64` is used. This parameter is ignored if `value` is already a Buffer.
`isRaw`	boolean	Indicates whether the data is in raw deflate format (without zlib headers). Default value is `false`.

Examples

To Base64-decode and inflate the value contained in a `_raw` field, you can use:

```
C.Decode.inflate(_raw)
```

Copy

To additionally specify a different encoding to use, run:

```
C.Decode.inflate(_raw, 'hex')
```

Copy

`C.Decode.mime()`

```
C.Decode.mime(val: string): string
```

Copy

Decode string that has been encoded with MIME RFC 2047 standard. This standard is frequently used in email subject lines within logs to handle non-ASCII characters.

Parameter	Type	Description
`val`	string	The input string to decode. This string must be formatted according to RFC 2047, typically starting and ending with `=?<charset>?<encodingType>?...?=`.

Examples

To decode a MIME-encoded string using the Quoted-Printable format, typically seen when the non-ASCII content is mostly readable text, use:

```
C.Decode.mime('=?UTF-8?Q?R=C3=A9sum=C3=A9?=') // Returns: 'Résumé'
```

Copy

To decode a MIME-encoded string using the Base64 format, often used when the non-ASCII content is complex or binary data, use:

```
C.Decode.mime('=?UTF-8?B?Q2Fmw6k=?=') // Returns: 'Café'.

### `C.Decode.uri()`

```js
C.Decode.uri(val: string): string
```

Copy

Performs URI-decoding of the given string.

Parameter	Type	Description
`val`	string	Value to decode.

C.Encode - Data Encoding Methods

`C.Encode.base64()`

```
C.Encode.base64(val: string | Buffer, trimTrailEq: boolean = false, encoding?: Buffer
```

Copy

Returns a Base64 representation of the given string or Buffer.
```

Parameter	Type	Description
`val`	any	Value to encode.
`trimTrailEq`	boolean	Whether to trim any trailing `=`.
`encoding`	string	Optional parameter that specifies the encoding of the value. Options include `base64`, `hex`, `utf-8`, and `binary`. The default is `utf-8`.

Examples

To Base64-encode the `username` field, and remove the trailing `=`, use:

```
C.Encode.base64(username, true)
```

Copy

To Base64-encode the `hexValue` field, where the value of the field is a hex string:

```
C.Encode.base64('4672616e6b7575757575', undefined, 'hex')
```

Copy

`C.Encode.deflate()`

```
C.Encode.deflate(value: Buffer | string, encoding: BufferEncoding | 'none' = 'base64'
```

Copy

Deflates and optionally Base64-encodes the supplied value.

Parameter	Type	Description
`value`	string or Buffer	Value to deflate.
`encoding`	string	Encoding to apply to the deflated result, for example: `base64`, `hex`, `utf-8`, `binary`, `none`. Default is `base64`. If `none` is specified, data will be returned as a Buffer.
`toRaw`	boolean	Indicates whether to return the raw deflated data, without zlib headers. Default is `false`, which will prepend normal zlib header bytes to the returned

Parameter	Type	Description
		value.

Examples

To deflate and hex-decode the value provided in an `inflated` field, run:

```
C.Encode.deflate(inflated, 'hex')
```

Copy

C.Encode.gzip()

```
C.Encode.gzip(value: Buffer | string, encoding?: string) : string | Buffer
```

Copy

Gzips, and optionally Base64-encodes, the supplied value.

Parameter	Type	Description
`value`	string	Value to gzip.
`encoding`	string	Encoding of `value`, for example: `'base64'`, `'hex'`, `'utf-8'`, `'binary'`, `'none'`. Default is `'base64'`. If `'none'` is specified, data will be returned as a Buffer.

Examples

To encode the `hostname` field, for example using `hex` encoding, run:

```
C.Encode.gzip(hostname, 'hex')
```

Copy

C.Encode.hex()

```
C.Encode.hex(val: string | number): string
```

Copy

Rounds the number to an integer and returns its hex representation (lowercase). If a string is provided, it will be parsed into a number or `NaN`.

Parameter	Type	Description
`val`	string | number	Value to convert to hex.

`C.Encode.mime()`

```
C.Encode.mime(val: string, encoding: string = 'Q', maxLineLength: number = 0, charset
```

Copy

Encode a string with MIME RFC 2047 standard. This method is used to prepare header values that contain non-ASCII characters for transport, such as email subject lines.

Parameter	Type	Description
`val`	string	The raw string to be encoded, including any non-ASCII characters.
`encoding`	MIME Encoding	Defines the encoding type. Use `Q` for Quoted-Printable, which is the default and can be omitted. Use `B` for Base64.
`maxLineLength`	number	Used for folding long lines, which is required by the RFC for headers. If greater than zero, inserts a newline delimiter after this defined number characters to ensure line wrapping. If `0`, no folding is performed.
`charset`	string	The character set of the input string, if it is not UTF-8.

Examples

To encode a MIME-encoded string using the Quoted-Printable format, use:

```
C.Encode.mime(_raw)
```

Copy

To encode a MIME-encoded string using the Base64 format, use:

```
C.Encode.mime(_raw, `B`)
```

Copy

To encode a string using Quoted-Printable and apply line folding to ensure no encoded line exceeds 20 characters:

```
C.Encode.mime('A much longer subject line with a non-ascii character: Español', 'Q',
// Returns `=?UTF-8?Q?A_much_longer_subject_line_with_a_non-ascii_characte?= \n =?UTF
```

Copy

C.Encode.uri()

```
C.Encode.uri(val: string): string
```

Copy

Returns the URI-encoded representation of the given string.

Parameter	Type	Description
val	string	Value to encode.

17.2.3. C.Lookup - Inline Lookup Methods

C.Lookup - Exact Lookup

```
C.Lookup: (file: string, primaryKey?: string, otherFields: string[]=[],ignoreCase: bo
```

Copy

Returns an instance of a lookup to use inline.

Examples

In this example, `host` is the name of the primary key field:

```
C.Lookup('lookup_name.csv', 'IP_field_name_in_lookup_file').match(host)
```

Copy

Here, the quoted `'event_field_or_string_to_match'` could be a string to match in the primary key field:

```
C.Lookup('name_of_lookup_file.csv', 'field_in_csv_to_match').match('event_field_or_st
```

Copy

This example checks whether `someValue` is present or not and returns the answer as a boolean.

```
C.Lookup('file', 'primaryKeyCol', additionalCols).match('someValue')
```

Copy

This expression returns a string - the `fieldToReturn` column from the matched lookup row:

```
C.Lookup('file', 'primaryKeyCol', additionalCols).match('someValue','fieldToReturn')
```

Copy

This expression returns an array - the specified `fieldToReturn1` and `fieldToReturn2` columns from the matched lookup row:

```
C.Lookup('file', 'primaryKeyCol').match('someValue', ['fieldToReturn1', 'fieldToRetu
```

Copy

To return an array with all columns from the matched lookup row, you can use:

```
C.Lookup('file', 'primaryKeyCol').match('someValue', [])
```

Copy

> `C.Lookup` can load lookup files of up to 10 MB.
>
> All inputs to Lookup methods' `match()` method must be strings. If your lookup file contains numeric fields, convert them to strings, e.g.: `.match(String(<fieldname>))`.
>
> You can use the optional `otherFields[]` argument, shown in the above `C.Lookup()` signatures and examples, to limit which columns of the lookup table will be available in a subsequent `.match()` call. If omitted, or set to `undefined`, all columns will be available.
>
> Cribl Edge 4.1 and newer support returning all columns from a matched row. Use the empty-array convention shown in the above examples: `.match('someValue',[])`.

`C.LookupCIDR` - CIDR Lookup

```
C.LookupCIDR: (file: string, primaryKey?: string, otherFields: string[]=[]) => Inline
```

Copy

Returns an instance of a CIDR lookup to use inline.

`C.LookupIgnoreCase` - Case-insensitive Lookup

```
C.LookupIgnoreCase: (file: string, primaryKey?: string, otherFields: string[]=[]) =>
```

Copy

Returns an instance of a lookup (ignoring case) to use inline. Works identically to `C.Lookup`, except ignores the case of lookup values. (Equivalent to calling `C.Lookup` with its fourth `ignoreCase?` parameter set to `true`).

C.LookupRegex - Regex Lookup

```
C.LookupRegex: (file: string, primaryKey?: string, otherFields: string[]=[]) => Inlir
```

Copy

Returns an instance of a Regex lookup to use inline.

C.Lookup.match()

```
InlineLookup.match(value: string): boolean

InlineLookup.match(value: string, fieldToReturn?: string): any
InlineLookup.match(value: string, fieldToReturn: string[]): {[key: string]: any}
```

Copy

Parameter	Type	Description
`value`	string	the value to look up.
`fieldToReturn`	string | string []	name of the lookup file > field to return.

Examples

To find the last row where the value from column `foo` matches the string `abc`, you can use the following expression. If a matching row is found, the expression returns the row's value for column `bar`.

```
C.Lookup('lookup-exact.csv', 'foo').match('abc', 'bar')
```

Copy

To find the last row where the value from column `transport` matches the string `tcp`, you can use the following expression. If a matching row is found, the expression returns the row's value for column `port_number`.

```
C.Lookup('service_names_port_numbers.csv', 'transport').match('tcp', 'port_number')
```

Copy

You can find the last row where the CIDR range in `cidr` includes `192.168.1.1` with the following expression. If a matching row is found, the expression returns the row's value for column `bar`.

```
C.LookupCIDR('lookup-cidr.csv', 'cidr').match('192.168.1.1', 'bar')
```

Copy

The following expression finds the last row where the value from column `cidr` matches `hostIP`. If a matching row is found, it returns the row's value for column `location`.

```
C.LookupCIDR('lookup-cidr.csv', 'cidr').match(hostIP, 'location')
```

Copy

To find the last row where the regex in column `foo` matches the string `manchester`, use the following expression. If a matching row is found, the expression returns the row's value for column `bar`.

```
C.LookupRegex('lookup-regex.csv', 'foo').match('manchester', 'bar')
```

Copy



With `C.LookupRegex`, ensure that your lookup file contains no empty lines - not even at the bottom. Any empty rows will cause `C.LookupRegex().match()` to always return `true`.

17.2.4. C.Mask - Data Masking Methods

The `C.Mask` expression provides a suite of methods for data masking, which is the process of obscuring sensitive information in your data. This ensures data privacy and maintains compliance with regulations like GDPR and HIPAA. It can protect personally identifiable information (PII) before sending it to a downstream Destination.

Use the methods in this expression to:

- **Redact sensitive information:** Replace data with a literal value like `REDACTED`. For example, replacing a credit card number like `1234-5678-9101-1121` to `REDACTED`.
- **Obfuscate data:** Replace values with a random or repeating character set. For example, obfuscating a username like `jdoe@example.com` by replacing it with a string of repeating characters, such as `XXXXXXXXXXXXXXX`.
- **Pseudonymize data:** Transform values into a non-identifiable, consistent value using a cryptographic hash. This technique allows you to perform analytics on the masked data without risking a privacy breach. For example, psudonymize a user ID like `user:1234567` into a consistent hash, such as `4b4f5d21c9a63c`. This makes it possible to analyze the data without revealing the original value.

By using `C.Mask` expressions, you can ensure that you handle sensitive data securely and that your Pipelines meet the necessary privacy and compliance standards.

For easier navigation, this page organizes the `C.Mask` methods by their purpose:

- Identity Validation and Masking: Methods for masking common identifiers like credit card and IMEI numbers, which also perform a basic validation check.
- Checksums and Validation: Methods that generate checksums and perform validation, which are the building blocks for more complex validation rules.
- Hashing: A collection of cryptographic hashing functions for pseudonymizing data while preserving its uniqueness for analytics.
- String Manipulation: General-purpose functions for creating random or repeating strings.

Identity Validation and Masking

`C.Mask.CC()`

Checks whether a value could be a valid credit card number. If the value matches the respective format, it masks a subset of the value. By default, all digits except the last 4 will be replaced with `X`.

Usage:

```
C.Mask.CC(value: string, unmasked: number = -4, maskChar: string='X'): string
```

Copy

Parameters:

Parameter	Type	Description
`value`	string	The string to mask, **if and only if** it could be a valid credit card number.
`unmasked`	number	Number of digits to leave unmasked: positive for left, negative for right, `0` for none.
`maskChar`	string	String to replace each digit with.

Examples:

Assuming your parsed data contains a `creditCard` field, you can mask this number by using a Mask Function. Apply it to the `cardNumber` field and create the following masking rule:

Match Regex	Replace Expression	Example result
`(.*)` (catches the whole field contents)	`${C.Mask.CC(g1)}`	`cardNumber: XXXXXXXXXX4860`

To control how the masking is performed, you can specify how many digit should be unmasked and what masking character to use:

Match Regex	Replace Expression	Example result
`(.*)` (catches the whole field contents)	`${C.Mask.CC(g1, 4, "Z")}`	`cardNumber: 6011ZZZZZZZZZZZZ`

`C.Mask.IMEI()`

Checks whether a value could be a valid IMEI number. If the value matches the respective format, it masks a subset of the value. By default, all digits except the last 4 will be replaced with `X`.

Usage:

```
C.Mask.IMEI(value: string, unmasked: number = -4, maskChar: string='X'): string
```

Copy

Parameters:

Parameter	Type	Description
`value`	string	The string to mask, **if and only if** it could be a valid IMEI number.
`unmasked`	number	Number of digits to leave unmasked: positive for left, negative for right, `0` for none.
`maskChar`	string	String to replace each digit with.

Examples:

Assuming your parsed data contains an `imei` field, you can mask this number by using a Mask Function. Apply it to the `imei` field and create the following masking rule:

Match Regex	Replace Expression	Example result
`(.*)` (catches the whole field contents)	`${C.Mask.IMEI(g1)}`	`imei: XXXXXXXXXXX7011`

To control how the masking is performed, you can specify how many digit should be unmasked and what masking character to use:

Match Regex	Replace Expression	Example result
`(.*)` (catches the whole field contents)	`${C.Mask.IMEI(g1, 4, "M")}`	`cardNumber: 5356MMMMMMMMMM`

`C.Mask.isCC()`

Checks whether the given value could be a valid credit card number, by computing the string's Luhn's checksum modulo 10 == `0`.

Usage:

```
C.Mask.isCC(value: string): boolean
```

Copy

Parameters:

Parameter	Type	Description
`value`	string	String to check for being a valid credit card number.

Examples:

The following example expression uses `isCC()` to check whether the `cardNumber` field actually contains a credit card number. If it does, the card number is replaced with the `REDACTED` string, otherwise, en empty string is returned.

```
C.Mask.isCC(cardNumber) ? C.Mask.REDACTED : ""
```

Copy

`C.Mask.isIMEI()`

Checks whether the given value could be a valid IMEI number, by computing the string's Luhn's checksum modulo 10 == `0`.

Usage:

```
C.Mask.isIMEI(value: string): boolean
```

Copy

Parameters:

Parameter	Type	Description
`value`	string	String to check for being a valid IMEI number.

Examples:

The following example expression uses `isIMEI()` to check whether the `imei` field actually contains an IMEI number number. If it does, the IMEI number is replaced with the `REDACTED` string, otherwise, en empty string is returned.

```
C.Mask.isIMEI(imei) ? C.Mask.REDACTED : ""
```

Copy

C.Mask.REDACTED

`C.Mask.REDACTED` is a literal string (not a function) that evaluates to `REDACTED`. It's used to completely remove sensitive data by replacing it with a clear, non-identifiable placeholder.

Usage:

```
C.Mask.REDACTED: string
```

Copy

Examples:

This example demonstrates using `C.Mask.REDACTED` within a conditional expression. It checks if the `cardNumber` field is a valid credit card number. If it is, the value is replaced with `REDACTED`. Otherwise, it keeps the original value.

```
C.Mask.isCC(cardNumber) ? C.Mask.REDACTED : cardNumber
```

Copy

Checksums and Validation

C.Mask.luhn()

Masks a value only if its checksum is valid. This is useful for sanitizing data and ensuring that only correctly formatted numbers are sent downstream.

It checks that value's Luhn checksum mod 10 is `0`, and masks a subset of the value. By default, all digits except the last 4 will be replaced with `X`. If the value's Luhn's checksum mod 10 is not `0`, then the value is returned unmodified.

Usage:

```
C.Mask.luhn(value: string, unmasked: number = -4, maskChar: string='X'): string
```

Copy

Parameters:

Parameter	Type	Description
`value`	string	The string to mask, **if and only if** the value's Luhn checksum mod 10 is `0`.
`unmasked`	number	Number of digits to leave unmasked: positive for left, negative for right, `0` for none.
`maskChar`	string	String to replace each digit with.

Examples:

Expression	Input Value	Output
`C.Mask.luhn(value)`	`49927398716` (a valid Luhn number)	`XXXXXXX8716`
`C.Mask.luhn(value)`	`49927398717` (an invalid Luhn number)	`49927398717`
`C.Mask.luhn(value, 4, "Y")`	`49927398716` (a valid Luhn number)	`4992YYYYYYY`

`C.Mask.luhnChecksum()`

Generates a checksum that can be used to validate a number. This is useful for building your own custom validation rules or for debugging.

Generates the Luhn checksum (used to validate certain credit card numbers, IMEIs, and more). By default, the mod 10 of the checksum is returned. Pass mod = `0` to get the actual checksum.

Usage:

```
C.Mask.luhnChecksum(value: string, mod: number=10): number
```

Copy

Parameters:

Parameter	Type	Description
`value`	string	String whose digits you want to perform the Luhn checksum on.
`mod`	number	Return checksum modulo for this number. If `0`, skip modulo. Default is `10`.

Examples:

This example demonstrates how to get the default `mod 10` checksum. This expression returns the last digit of the checksum, which is commonly used to validate a number. A valid Luhn number will return `0`.

```
// This will return the number 0
C.Mask.luhnChecksum("49927398716")
```

Copy

This example demonstrates how to get the full checksum. You can get the full checksum by setting `mod` to `0`.

```
// This will return the number 70, which is the full checksum
C.Mask.luhnChecksum("49927398716", 0)
```

Copy

Hashing

`C.Mask.crc32()`

Generates a CRC32 hash of given value.

Usage:

```
C.Mask.crc32(value: string): string
```

Copy

Parameter:

Parameter	Type	Description
`value`	string	The string to hash.

Examples:

Expression	Input Value	Output
`C.Mask.crc32(value)`	`example_log_123`	`cf0e3d3a`
`C.Mask.crc32(host + file_path)`	`webserver01/var/log/app.log`	`b9b661b6`

C.Mask.md5()

Generates an MD5 hash of a given value.



`C.Mask.md5()` doesn't work when Cribl Stream is running in FIPS mode.

Usage:

```
C.Mask.md5(value: string, len?: string | number, encoding?: BufferEncoding): string
```

Copy

Parameters:

Parameter	Type	Description
`value`	string	The string to hash.
`len`	string or number	Length of hash to return: `0` for full hash, a +number for left or a -number for right substring. If a string is passed, its length will be used.
`encoding`	string	Optional parameter that specifies the encoding of the value. Options include `base64`, `hex`, `utf-8`, and `binary`. The default is `utf-8`.

Examples:

To hash the contents of a `username` field with the MD5 algorithm, run:

```
C.Mask.md5(username)
```

Copy

To hash the contents of an `authToken` field, where the value of the field is a base64-encoded string, with the MD5 algorithm, run:

```
C.Mask.md5(authToken, undefined, 'base64')
```

Copy

C.Mask.sha1(), C.Mask.sha256(), C.Mask.sha512(), C.Mask.sha3_256(), C.Mask.sha3_512()

Generates a hash of a given value. Supported algorithms:

- SHA1
- SHA256
- SHA512
- SHA3-256
- SHA3-512

Usage:

```
C.Mask.sha1(value: string, len?: string | number, encoding?: BufferEncoding): string
C.Mask.sha256(value: string, len?: string | number, encoding?: BufferEncoding): strir
C.Mask.sha512(value: string, len?: string | number, encoding?: BufferEncoding): strir
C.Mask.sha3_256(value: string, len?: string | number, encoding?: BufferEncoding): str
C.Mask.sha3_512(value: string, len?: string | number, encoding?: BufferEncoding): str
```

Copy

Parameters:

Parameter	Type	Description
`value`	string	The string to hash.
`len`	string or number	Length of hash to return: `0` for full hash, a +number for left, or a -number for right substring. If a string is passed, its length will be used.
`encoding`	string	Optional parameter that specifies the encoding of the value. Options include `base64`, `hex`, `utf-8`, and `binary`. The default is `utf-8`.

Examples:

To hash the contents of a `secret` field with the SHA256 algorithm, run:

```
C.Mask.sha256(secret)
```

Copy

To hash the contents of an `authToken` field, where the value of the field is a base64-encoded string, with the SHA256 algorithm, run:

```
C.Mask.sha256(authToken, undefined, 'base64')
```

Copy

String Manipulation

`C.Mask.random`

Generates a random alphanumeric string.

Usage:

```
C.Mask.random(len?: string | number): string
```

Copy

Parameters:

Parameter	Type	Description
`len`	string or number	Length of the result; or, if a string, use its length.

Examples:

Assuming you have some identifying user information in a `username` field, you can replace it with a random string of the same length with:

```
C.Mask.random(username)
```

Copy

Input	Output
`McGoatFace`	`xY2KLHkyTg`

`C.Mask.repeat()`

Generates a repeating character or string pattern, such as `XXXX`.

Usage:

```
C.Mask.repeat(len?: number | string, char: string = 'X'): string
```

Copy

Parameters:

Parameter	Type	Description
`len`	string or number	Length of the result; or, if a string, use its length.
`char`	string	Pattern to repeat `len` times.

Examples:

The following example will repeat the `GOAT` string as many time as there are characters in the first parameter, `goat`, so four times:

```
C.Mask.repeat("goat", "GOAT")
```

17.2.5. C.Net - Network Methods

C.Net.cidrMatch()

```
C.Net.cidrMatch(cidrIpRange: string, ipAddress: string): boolean
```

Copy

Determines whether the supplied IPv4 or IPv6 `ipAddress` is inside the range of addresses identified by `cidrIpRange`.

Parameter	Type	Description
`cidrIpRange`	string	IP address range in CIDR format. E.g., `10.0.0.0/24`.
`ipAddress`	string	The IP address to test for inclusion in `cidrIpRange`.

Examples

(IPv4): `C.Net.cidrMatch('10.0.0.0/24', '10.0.0.100')` returns `true`.

(IPv6): `C.Net.cidrMatch('2001:db8::/32', '2001:db8:0:0:0:ffff:0:0')` returns `true`.

C.Net.communityIDv1()

```
C.Net.communityIDv1(sourceIP: string, destIP: string, sourcePort: number | undefined,
```

Copy

Calculates the Community ID for a given network flow. A Community ID is a unique identifier that is reversible - it is the same for both sides of a connection. It is calculated based on source and destination IP addresses, ports, a protocol, and (optionally) a seed (explained below). When the protocol is ICMP, the Community ID will be reversible provided that the network flow uses types with a clear request/response relationship (for example, Echo and Echo Reply).

Usage

This method returns a Base64-encoded string.

Provide IP addresses - whether IPv4 or IPv6 - as strings.

When the protocol is ICMP (v4 or v6), `sourcePort` corresponds to message type and `destPort` corresponds to message code. For other Layer 3 protocols, `sourcePort` and `destPort` should be `undefined`.

You can specify `proto` either as a number (recommended, to avoid ambiguity) or a string.

- If a string, `proto` must be the desired protocol's keyword as specified in the IANA protocol registry.
- For network flows coming from the Cribl Stream NetFlow & IPFIX Source, use the protocol name as-is - the Source will provide it in correct form.

The optional `seed` parameter defaults to `0`. If you want to generate different Community IDs for different use cases within the same flow, change `seed` to a different value for each use case.

Parameter	Type	Description
sourceIP	string	Source IPv4 or IPv6 address
destIP	string	Destination IPv4 or IPv6 address
sourcePort	number or undefined	Source port. For ICMP, use the ICMP type. For other portless protocols, use `undefined`.
destPort	number or undefined	Destination port. For ICMP, use the ICMP code. For other portless protocols, use `undefined`.
proto	number or string	IP protocol number, such as `6`, `17`, or `1`. May also be the protocol name, such as `'tcp'`, `'udp'`, or `'icmp'`.
seed	number	Optional seed value; defaults to `0`

Examples

Bidirectional TCP flow using the protocol number.

```
// Protocol number 6 is TCP

C.Net.communityIDv1('1.2.3.4','5.6.7.8',1122,3344,6)
// '1:wCb3OG7yAFWelaUydu0D+125CLM='

C.Net.communityIDv1('5.6.7.8','1.2.3.4',3344,1122,6)
// '1:wCb3OG7yAFWelaUydu0D+125CLM='
```

Copy

Bidirectional UDP flow using the protocol name.

```
C.Net.communityIDv1('1.2.3.4','5.6.7.8',1122,3344,'udp')
// '1:0Mu9InQx6z4ZiCZM/7HXi2WMhOg='

C.Net.communityIDv1('5.6.7.8','1.2.3.4',3344,1122,'udp')
// '1:0Mu9InQx6z4ZiCZM/7HXi2WMhOg='
```

Copy

ICMP Echo/Echo Reply flow.

```
// Protocol number 1 is ICMP
// Echo is Type=8, Code=0
// Echo Reply is Type=0, Code=0

// Echo (equivalent of request)
C.Net.communityIDv1('1.2.3.4','5.6.7.8',8,0,1)
// '1:crodRHL2FEsHjbv3UkRrfbs4bZ0='

// Echo Reply (equivalent of response)
C.Net.communityIDv1('5.6.7.8','1.2.3.4',0,0,1)
// '1:crodRHL2FEsHjbv3UkRrfbs4bZ0='
```

Copy

C.Net.isIp()

```
C.Net.isIp(value: string): boolean
```

Copy

Determines whether the value supplied is an IP address (either IPv4 or IPv6).

Parameter	Type	Description
value	string	The IP address to test.

C.Net.isIpV4()

```
C.Net.isIpV4(value: string): boolean
```

Copy

Determines whether the value supplied is an IPv4 address.

Parameter	Type	Description
value	string	The IP address to test.

C.Net.isIpV6()

```
C.Net.isIpV6(value: string): boolean
```

Copy

Determines whether the value supplied is an IPv6 address.

Parameter	Type	Description
value	string	The IP address to test.

C.Net.isIPV4Cidr()

```
C.Net.isIPV4Cidr(value: string): boolean
```

Copy

Determines whether a value supplied is an IPv4 CIDR range.

Returns `true` if the value is a valid IPv4 CIDR range; otherwise, `false`.

Parameter	Type	Description
`value`	string	String value that may be an IPv4 CIDR range.

C.Net.isIPV6Cidr()

```
C.Net.isIPV6Cidr(value: string): boolean
```

Copy

Determine whether a value supplied is an IPv6 CIDR range.

Returns `true` if the value is a valid IPv6 CIDR range; otherwise, `false`.

Parameter	Type	Description
`value`	string	String value that may be an IPv6 CIDR range.

C.Net.isIpV4AllInterfaces()

```
C.Net.isIpV4AllInterfaces(value): boolean
```

Copy

Determines whether the value supplied is the IPv4 all-interfaces address, typically used to bind to all IPv4 interfaces - i.e., '0.0.0.0'.

Parameter	Type	Description
`value`	any	The IP address to test.

C.Net.isIpV6AllInterfaces()

```
C.Net.isIpV6AllInterfaces(value): boolean
```

Copy

Determines whether the value supplied is an IPv6 all-interfaces address, typically used to bind to all IPv6 interfaces - one of: '::', '0:0:0:0:0:0:0:0', or '0000:0000:0000:0000:0000:0000:0000:0000'.

Parameter	Type	Description
`value`	any	the IP address to test.

C.Net.ipv6Normalize()

```
C.Net.ipv6Normalize(address: string): string
```

Copy

Normalizes an IPV6 address based on RFC draft-ietf-6man-text-addr-representation-04.

Parameter	Type	Description
`address`	string	The IPV6 address to normalize.

C.Net.isPrivate()

```
C.Net.isPrivate(address: string): boolean
```

Copy

Determines whether the supplied IPv4 address is in the range of private addresses per RFC1918.

Parameter	Type	Description
`address`	string	The IP address to test.

C.Net.parseAddressOrRangeString()

```
C.Net.parseAddressOrRangeString(source: string, mask?: string | number)
```

Copy

Parses a string representation of an IP or IP range.

Returns an `IPv4` or `IPv6` object. When the `source` argument is an IP range, this method returns the first IP in the range.

Parameter	Type	Description
`source`	string	The string to parse.
`mask`	string | number	Optional parameter containing the CIDR mask. Used only when `source` is a valid IP string.

C.Net.IPv4_CIDR_REGEX

17.2.6. C.Text - Text Methods

C.Text.entropy()

```
C.Text.entropy(bytes: Buffer | string): number
```

Copy

Computes the Shannon entropy of the given Buffer or string.

Returns the entropy value; or `-1` in case of an error.

Parameter	Type	Description
`bytes`	Buffer | string	value to undergo Shannon entropy computation.

C.Text.hashCode()

```
C.Text.hashCode(val: string | Buffer | number): number
```

Copy

Computes hashcode (djb2) of the given value.

Returns hashcode value.

Parameter	Type	Description
`val`	string | Buffer | number	value to be hashed.

C.Text.isASCII()

```
C.Text.isASCII(bytes: Buffer | string): boolean
```

Copy

Checks whether all bytes or chars are in the ASCII printable range.

Returns `true` if all chars/bytes are within ASCII printable range; otherwise, `false`.

Parameter	Type	Description
`bytes`	string | Buffer	value to check for character range.

`C.Text.isUTF8()`

```
C.Text.isUTF8(bytes: Buffer | string): boolean
```

Copy

Checks whether the given Buffer contains valid UTF8.

Returns `true` if bytes are UTF8; otherwise, `false`.

Parameter	Type	Description
`bytes`	Buffer | string	bytes to check.

`C.Text.parseWinEvent()`

```
C.Text.parseWinEvent(xml: string, nonValues: string[] = C.Text._WIN_EVENT_NON_VALUES)
```

Copy

Parses an XML string representing a Windows event into a compact, prettified JSON object. Works like `C.Text.parseXml`, but with Windows events, produces more-compact output. For a usage example, see Reducing Windows XML Events.

Returns an object representing the parsed Windows Event; or `undefined` if the input could not be parsed.

Parameter	Type	Description
`xml`	string	an XML string; or an event field containing the XML.

Parameter	Type	Description
`nonValues`	string[]	array of string values. Elements whose value equals any of the values in this array will be omitted from the returned object. Defaults to `['-']`, meaning that elements whose value equals `-` will be discarded.

C.Text.parseXml()

```
C.Text.parseXml(xml: string, keepAttr: boolean = true, keepMetadata: boolean = false,
```

Copy

Parses an XML string and returns a JSON object. Can be used with Eval Function to parse XML fields contained in an event, or with ad hoc XML.

Returns an object representing the parsed XML; or `undefined` if the input could not be parsed. An input collection of elements will be parsed into an array of objects.

Parameter	Type	Description
`xml`	string	XML string, or an event field containing the XML.
`keepAttr`	boolean	whether or not to include attributes in the returned object. Defaults to `true`.
`keepMetadata`	boolean	whether or not to include metadata found in the XML. The `keepAttr` parameter must be set to `true` for this to work. Defaults to `false`. (Eligible metadata includes namespace definitions and prefixes, and XML declaration attributes such as encoding, version, etc.)
`nonValues`	string[]	array of string values. Elements whose value equals any of the values in this array will be omitted from the returned object. Defaults to `[]` (empty array), meaning discard no elements.

C.Text.relativeEntropy()

```
C.Text.relativeEntropy(bytes: Buffer | string, modelName: string = 'top_domains'): nu
```

Copy

Computes the relative entropy of the given Buffer or string.

Returns the relative entropy value, or `-1` in case of an error.

Parameter	Type	Description
`bytes`	Buffer | string	Value whose relative entropy to compute.
`modelName`	string	Optionally, override the default `$CRIBL_HOME/data/lookups/model_relative_entropy_top_domains.csv` model used to test the input. Create a custom lookup file with the same column and value structure as the default, and store it in the same path, as `model_relative_entropy_<custom-name>.csv`. To reference it, pass your `<custom-name>` substring as the `modelName` parameter.

When using `modelName` in a distributed deployment, the corresponding paths are `$CRIBL_HOME/groups/<worker-group-name>/data/lookups/`. Creating your custom lookup file via Cribl Edge's UI will automatically set the appropriate paths.

17.2.7. C.Time - Time Methods

C.Time.adjustTZ()

```
C.Time.adjustTZ(epochTime: number, tzTo: string, tzFrom: string = 'UTC'): number | ur
```

Copy

Adjusts a timestamp from one timezone to another.

Returns the adjusted timestamp, in Unix time (ms).

Parameter	Type	Description
`epochTime`	number	Timestamp to adjust, in Unix time.
`tzTo`	string	Timezone to adjust to (full name).
`tzFrom` (optional)	string	Timezone of the timestamp.

Cribl relies on this list of TZ Database Time Zones.

Note that these are timezone identifiers (full names), not abbreviations such as “CET”, “PST”, “EST”, and so on.

Examples

To adjust a timestamp found in the `_time` field to a selected timezone (here, Australia/Melbourne), use:

```
C.Time.adjustTZ(_time, "Australia/Melbourne")
```

Copy

The method returns the timestamp in milliseconds. If you want to further operate on it using methods such as `strftime()`, which takes as parameter Unix time in seconds, you can divide it by 1000:

```
C.Time.strftime(C.Time.adjustTZ(_time, "Australia/Melbourne") / 1000, "%d %B %Y %H %M
```

Copy

C.Time.clamp()

```
C.Time.clamp(date: T, earliest: T, latest: T, defaultDate?: T ): T  | undefined
```

Copy

Constrains a parsed timestamp to realistic earliest and latest boundaries.

Returns a JavaScript Date object or timestamp, depending on the format of the `date` parameter.

Parameter	Type	Description
`date`	Date | number	Timestamp to constrain, in Unix time (ms) or JavaScript Date format.
`earliest`	Date | number	Earliest allowable timestamp, in Unix time (ms) or JavaScript Date format.
`latest`	Date | number	Latest allowable timestamp, in Unix time (ms) or JavaScript Date format.
`defaultDate` (optional)	Date | number	Default date, in Unix time (ms) or JavaScript Date format. This date is used for values that fall outside the `earliest` or `latest` boundaries.

Examples

Suppose you have a date in a `"YYYY-MM-DD"` format in a `time` field. To make sure this date does not exceed the range of 1 Jan 2020 - 1 Jan 2025, you can use:

```
C.Time.clamp(time, "2020-01-01", "2025-01-01")
```

Copy

Input	Output
`"2016-02-11"`	`"2020-01-01"` (set to the earliest value)

Input	Output
`"2026-01-01"`	`"2025-01-01"` (set to the latest value)
`"2023-12-24"`	`"2023-12-24"` (unchanged, because it fits inside the defined range)

To make sure that all dates that fall outside the bounds are changed to a specific value, add the third parameter:

```
C.Time.clamp(time, "2020-01-01", "2025-01-01", "2022-02-22")
```

Copy

Input	Output
`"2016-02-11"`	`"2022-02-22"` (set to the default value)
`"2026-01-01"`	`"2022-02-22"` (set to the default value)
`"2023-12-24"`	`"2023-12-24"` (unchanged, because it fits inside the defined range)

`C.Time.strftime()`

```
C.Time.strftime(date: number | Date | string, format: string, utc: boolean = true): s
```

Copy

Formats a Date object or timestamp number as a time string, using d3js time format.

Returns representation of the given date in the specified format.

Parameter	Type	Description
`date`	number | Date | string	Date to format, in Unix time (in seconds) or JavaScript Date format.
`format`	string	New format for the date.
`utc` (optional)	boolean	Whether to output the time in UTC (`true`), rather than in local timezone.

Examples

To transform a `_time` field containing a timestamp (in Unix time) to a string with time formatted, for example: `15 November 2023, 08:03`, you can use:

```
C.Time.strftime(_time, "%d %B %Y, %H:%M")
```

Copy

Some common formats (assuming timestamp in Unix time of 1702460134) are:

Format	Output
`"%Y-%m-%d"` (ISO date)	`"2023-12-13"`
`"%d %B %Y %H:%M:%S"`	`"13 December 2023 09:35:34"`
`"%x %X"` (the locale's date and time format)	`"12/13/2023 9:35:34 AM"`

For detailed reference of the date and time tokens, see d3js reference or Auto Timestamp.

C.Time.strptime()

```
C.Time.strptime(str: string, format: string, utc: boolean = true, strict: boolean = f
```

Copy

Extracts time from a string using d3js time format that defines the format of the incoming string.

Returns a parsed JavaScript Date object, or `null` if the specified `format` did not match.

Parameter	Type	Description
`str`	string	Timestamp to parse.
`format`	string	Format to parse.
`utc` (optional)	boolean	Whether to interpret times as UTC (`true`), rather than as local time.

Parameter	Type	Description
`strict` (optional)	boolean	Whether to return `null` if there are any extra characters after timestamp.

Examples

To convert a string in the `"YYYY-MM-DD"` format, for example `"2023-12-12"`, to a Date object, use:

```
C.Time.strptime(F, "%Y-%m-%d")
```

Copy

Additionally, to interpret the time in UTC, and make sure null is returned if the timestamp contains any trailing characters, use:

```
C.Time.strptime(F, "%Y-%m-%d", true, true)
```

Copy

For detailed reference of the date and time tokens, see d3js reference or Auto Timestamp.

C.Time.timestampFinder()

```
C.Time.timestampFinder(utc?: boolean).find(str: string): IAutoTimeParser
```

Copy

Extracts time from the specified field, using the same algorithm as the Auto Timestamp Function and Event Breakers.

Returns representation of the extracted time; truncates timestamps to three-digit (milliseconds) resolution, omitting trailing zeros.

Parameter	Type	Description
`utc`	boolean	Whether to output the time in UTC (`true`), rather than in local timezone.
`str`	string	The field in which to search for the time.

Examples

To find timestamps in the `_raw` field and output them in UTC, you can use:

```
C.Time.timestampFinder(true).find(_raw)
```

Copy

You can then format the string output with `strftime()`:

```
C.Time.strftime(C.Time.timestampFinder(true).find(_raw), "%d.%m.%Y, %H.%M")
```

Copy

C.Time.timePartition()

```
C.Time.timePartition(date: number | Date | string, level: 'd' | 'h'): string | undefi
```

Copy

Provides the part of the file path prefix that contains the date.

Returns a string in the format `YYYY/MM/DD` for `d` and `YYYY/MM/DD/HH` for `h`.

Parameter	Type	Description
`date`	number | Date | string	Datetime to extract the file path prefix for.
`level`	string	Granularity of the partitioning, `d` for daily, and `h` for hourly.

Examples

To get the file path prefix from the `_time` field:

```
C.Time.timePartition(_time)
```

Copy

This will return the prefix in the form of, for example: `2024/04/23`.

C.Time.s3TimePartition()

```
C.Time.s3TimePartition(date: number | Date | string, level: 'd' | 'h'): string | unde
```

Copy

Provides the full S3 file path prefix for a given datetime.

Returns the file path prefix as a string.

Parameter	Type	Description
`date`	number | Date | string	Datetime to extract the file path prefix for.
`level`	string	Granularity of the partitioning, `d` for daily, and `h` for hourly.

Examples

To get the S3 file path prefix from the `_time` field:

```
C.Time.s3TimePartition(_time)
```

Copy

17.2.8. Miscellaneous Expression Methods

C.Schema - Schema Methods

`C.Schema`

```
C.Schema: (id: string) => SchemaValidator
```

Copy

`C.Schema.validate()`

```
C.SchemaValidator.validate(data: any): boolean
```

Copy

Validates the given object against the schema.

Returns `true` when schema is valid; otherwise, `false`.

Parameter	Type	Description
`data`	any	Object to be validated.

Examples

To validate whether a `myField` conforms to `schema1`, you can use:

```
C.Schema('schema1').validate(myField)
```

Copy

See Schema Library for more details.

C.Secret - Secrets-Management Methods

`C.Secret()`

```
C.Secret: (id: string, type?: string): ISecret
C.Secret(id: string, type: 'keypair') => IPairSecret
C.Secret(id: string, type: 'text') => ITextSecret
C.Secret(id: string, type: 'credentials') => ICredentialsSecret
```

Copy

Returns a secret matching the specified ID.

Parameter	Type	Description
`id`	string	ID of the secret.
`type` (optional)	'text' | 'keypair' | 'credentials'	Type of the secret.

Examples

To return a text secret with ID `victorias` (or with undefined, if no such secret exists), use:

```
C.Secret('victorias', 'text')
```

Copy

You can also get attributes of secrets with the following expressions:

```
C.Secret('api_key', 'keypair').secretKey
C.Secret('secret_hash', 'text').value
C.Secret('user_pass', 'credentials').password
```

Copy

Common returned attributes for `ISecret` objects:

- `secretType` - one of `keypair`, `text`, or `credentials`.

- `description` (optional) - the secret description.
- `tags` (optional) - a comma separated list of tags.

Additional returned attributes for `IPairSecret` objects:

- `apiKey` - the API key value
- `secretKey` - the Secret key value

Additional returned attributes for `ITextSecret` objects:

- `value` - the text value

Additional returned attributes for `ICredentialsSecret` objects:

- `username` - the username value
- `password` - the password value

See Securing Cribl Stream > Secrets for more details.

C.env - Environment

`C.env`

```
C.env: {[key: string]: string;}
```

Copy

Returns an object containing Cribl Edge's environment variables. All internal variables, that is, those starting with `CRIBL_`, are accessible in this method.

Examples

To return the parent of Cribl's `bin` directory (generally `/opt/cribl`), use:

```
C.env.CRIBL_HOME
```

Copy

To return the hostname of the machine where Cribl Edge is running, use:

```
C.env.HOSTNAME
```

Copy

C.os - System Methods

`C.os.hostname()`

Returns hostname of the system running this Cribl Edge instance.

C.vars - Global Variables

See Global Variables Library for more details.

> Fleet variables are designed specifically for configuration binding. They are not accessible using the `C.vars` expression in Pipeline Functions. You can only bind these variables to Source and Destination configuration fields that display the **Variable** icon ◈.
>
> The only exception is the Redis Function, which can resolve Fleet variables at runtime to manage external state or lookups.

C.version - Cribl Edge Versions

`C.version`

Returns the Cribl Edge version currently running.

`C.confVersion`

Returns the commit hash of the Edge Node's current config version. (Evaluates only against live data sent through Edge Nodes. Values will be undefined in the Leader's Preview pane.)

C.Misc - Miscellaneous Utility Methods

`C.Misc.zip()`

```
C.Misc.zip(keys: string[], values: any[], dest?: any): any
```

Copy

Sets the given keys to the corresponding values on the given `dest` object. If `dest` is not provided, a new object will be constructed.

Returns object on which the fields were set.

Parameter	Type	Description
`keys`	string[]	Field names corresponding to keys.
`values`	any[]	Values corresponding to values.
`dest`	any	Object on which to set field values.

Examples

Let's take the following expression:

```
people = C.Misc.zip(titles, names)
```

Copy

If sample data contains: `titles=['ceo', 'svp', 'vp']`, `names=['foo', 'bar', 'baz']`, this expression create an object called `people`, with key names from elements in `titles`, and with corresponding values from elements in `names`.

Result: `"people": {"ceo": "foo", "svp": "bar", "vp": "baz"}`

`C.Misc.uuidv4()`

```
C.Misc.uuidv4(): string
```

Copy

Returns a version 4 (random) UUID in accordance with RFC-4122.

Examples

To create a UUIDv4, use:

```
C.Misc.uuidv4()
```

Copy

Result: a conforming UUIDv4, such as `58d8be36-4db0-4b1c-ac80-28bb03c45e0d`. It is highly improbable that two version 4 UUIDs will ever have the same value.

`C.Misc.uuidv5()`

```
C.Misc.uuidv5(name: string, namespace: string): string
```

Copy

Returns a version 5 (namespaced) UUID in accordance with RFC-4122 for the given `name` and `namespace`. If `namespace` is not a valid UUID, this function will fail.

Parameter	Type	Description
`name`	string	Any arbitrary name to use in UUID generation.
`namespace`	string	One of `DNS`, `URL`, `OID`, or `X500` to use a predefined namespace, or else a valid UUID.

Examples

To create a UUIDv5 with the predefined DNS namespace, use:

```
C.Misc.uuidv5('example', 'DNS')
```

Copy

Result: a UUID that is highly likely to be the same for the same name and namespace. In this case, the result would be `7cb48787-6d91-5b9f-bc60-f30298ea5736`.

C.Misc.validateUUID()

```
C.Misc.validateUUID(maybeUUID: string): boolean
```

Copy

Returns `true` if `maybeUUID` is a valid UUID of any version, and false otherwise.

Parameter	Type	Description
`maybeUUID`	string	A string to test for a valid UUID.

Examples

```
C.Misc.validateUUID(C.Misc.uuidv4())
```

Copy

Result: returns `true`

```
C.Misc.validateUUID('clearly not a UUID')
```

Copy

Result: returns `false`

C.Misc.getUUIDVersion()

```
C.Misc.getUUIDVersion(uuid: string): number
```

Copy

Returns a `number` of the UUID version given by `uuid` if it is a valid UUID, otherwise `undefined`.

Parameter	Type	Description
`uuid`	string	A UUID for which to determine the version.

Examples

```
C.Misc.getUUIDVersion(C.Misc.uuidv4())
```

Copy

Result: `4`

```
C.Misc.getUUIDVersion(C.Misc.uuidv5('example', 'X500'))
```

Copy

Result: `5`

`C.WorkerGroupId`

```
C.WorkerGroupId: string
```

Copy

Returns the name of the current Fleet. Use this expression to enrich logs or metrics with the Fleet identity, enabling filtering and grouping in dashboards and analytics tools. This method always returns a string, even in single-node deployments.



Prefer `C.WorkerGroupId` over `C.env.CRIBL_GROUP_ID` where possible. The environment variable may be unset or stale, whereas `C.WorkerGroupId` always reflects the current Fleet assignment from your configuration.

Example

To add the current Fleet name to an event field:

17.3. CLI Reference

Command line interface basics and reference

In addition to starting and stopping the Cribl Edge server, the Cribl Edge command line interface enables you to initiate many configuration and administrative tasks directly from your terminal.



The command-line interface is only available to on-prem or hybrid customers.

To use the CLI:

1. First check that you have read/write access to the directory where Cribl Edge is installed. Contact your Cribl administrator to request access if needed.
2. Navigate to the directory where Cribl Edge is installed:

```
cd $CRIBL_HOME/bin
```

Copy

3. Execute a command. CLI commands have this basic syntax:

```
./cribl <command> <sub-command> <options> <arguments>
```

Copy



Not all commands have sub-commands.

4. To see the documentation for any command, append the `--help` option. For example:

```
./cribl vars --help

./cribl vars get --help

./cribl vars get -i myArray --help
```

Copy

Prevent and Troubleshoot CLI Problems

Immediate Execution

As indicated in the command reference and sample output, some commands take effect immediately.

Commands that require additional input will echo the sub-commands, options, and arguments they expect.

Persistent Volumes

If you start Cribl Edge with the `CRIBL_VOLUME_DIR` variable, all subsequent CLI commands should have this variable defined. Otherwise, those commands will apply the Cribl Edge default directories, yielding misleading results.

You can set `CRIBL_VOLUME_DIR` as an environment variable, or you can explicitly include it in each command, as in this example:

```
CRIBL_VOLUME_DIR=<writable-path-name> /opt/cribl/bin/cribl status
```

When set, `$CRIBL_VOLUME_DIR` overrides `$CRIBL_HOME`.

Avoid setting `$CRIBL_VOLUME_DIR` to an existing folder because it creates predefined folders in the specified directory. If that directory already contains folders with those names, Cribl Edge will overwrite those directories.

Failover Mode

In failover mode, set the `CRIBL_CONF_DIR` variable to your failover directory, ensuring your CLI commands access the correct configuration directory. For example, `CRIBL_CONF_DIR=/<path_to_failover_volume> cribl diag create`.

Your failover directory is either defined by the environment variable `CRIBL_DIST_LEADER_FAILOVER_VOLUME` (alternatively `CRIBL_DIST_MASTER_FAILOVER_VOLUME`), or defined in the instance.yml configuration file.

Commands Available

To see a list of available commands, enter `./cribl` alone (or the equivalent `./cribl help`). To execute a command, or to see its required parameters, enter `./cribl <command>`.

Command	Description
auth	Command to log into or out of the product
boot-start	Command to enable or disable product boot-start
cloud-workspace	Command to update a Leader with new config that allows it to connect to the Cribl.Cloud Leader and send it usage metrics
decrypt	Command to decrypt data with a secret key
diag	Command to manage diagnostic bundles
encrypt	Command to encrypt data with a secret key
git	Command to manage Worker Group or Fleet configuration
help	Displays a list of commands and their help
keys	Command to manage encryption keys
limits	Command to control the availability of Cribl features
mode-master	Command to configure an instance as a Leader
mode-single	Command to configure an instance as a Single-instance deployment
mode-edge	Command to configure Cribl Edge as a Single-instance deployment
mode-worker	Command to configure Cribl Stream as a Worker instance
mode-managed-edge	Command to configure Cribl Edge as an Edge Node
mode-outpost	Command to configure an instance as a Cribl Outpost
nc	Command to listen a port for traffic and output stats and data
node	Command to execute a JavaScript file
pack	Command to mange Cribl packs
parquet	Command to view a Parquet file, its metadata, or its schema
pipe	Command to feed stdin to a Pipeline
pq	Command for persistent queue performance benchmarking
reload	Command to reload the product

Command	Description
restart	Command to restart the product
start	Command to start the product
status	Command to display the product status
stop	Command to stop the product
vars	Command to manage global variables

17.3.1. Auth

Log into or out of Cribl Edge.

Sub-commands:

- `auth login`
- `auth logout`
- `auth mf`

Sub-commands and Options

`auth login`

Login with a username and password or use a file with credentials.

Usage:

```
./cribl auth login -H http://localhost:9000 -u example-username -p example-password
./cribl auth login -H http://localhost:9000 -f example-credentials-file.txt
```

Copy

See additional `auth login` Examples.

Arguments:

Option	Definition
`-H <host>`	Host URL. Example: `http://localhost:9000`.
`-u <username>`	Username.
`-p <password>`	Password.
`-f <file>`	File with credentials.

Arguments of the `login` subcommand correspond to the following environment variables:

Argument	Variable
`<username>`	`CRIBL_USERNAME`
`<password>`	`CRIBL_PASSWORD`
`<host>`	`CRIBL_HOST`

`auth login` Examples

Launch interactive login:

```
$CRIBL_HOME/bin/cribl auth login
```

Append credentials as command arguments:

```
$CRIBL_HOME/bin/cribl auth login -h <url> -u <username> -p <password>
```



All `-h` and `host` arguments are optional, provided that the `cribl.yml` file's `api:` section lists the API host and port.

Provide credentials in environment variables:

```
CRIBL_HOST=<url> CRIBL_USERNAME=<username> CRIBL_PASSWORD=<password>
$CRIBL_HOME/bin/cribl auth login
```

Provide credentials in a file:

```
$CRIBL_HOME/bin/cribl auth login -f <path/to/file>
```

Corresponding file contents:

```
host=<url>
username=<username>
password=<password>
```

Copy

`auth logout`

Logout from the current session.

Usage:

```
./cribl auth logout
```

Copy

auth mf

Control multi-factor authentication (PIV).

Usage:

```
./cribl auth mf -e true
```

Copy

See additional auth mf Examples.

Arguments:

Option	Definition
-e <enable>	Disable or enable multi-factor authentication using PIV. Accepts: [“true”, “false”].
-u <usernameRegex>	Username regex pattern (defaults to ‘/CN=(?:\w+.)+(\d+)/’).
-f <usernameField>	Field from which username is to be extracted (defaults to ‘subject’). Accepts: [subject,san]
-o <accessControlAllowOrigin>	Access control allow origin header. Example: “https://127.0.0.1:9000” or “https://fqdn”
-a <apiServerUrl>	API server internal FQDN or IP:PORT combo. Make sure this is also part of server TLS certificate SAN->IP or SAN->DNS (such as “https://[IP]:[PORT]” or “https://internal-fqdn”).

auth mf Examples

Check the status of multi-factor authentication using Personal Identity Verification (PIV):

```
$CRIBL_HOME/bin/cribl auth mf
```

The response includes the status (disabled or enabled) and the current configuration, along with notifications for unconfigured values and issues like failure to restart or enable TLS. For example:

```
PIV multi-factor authentication is disabled.
Username field not defined. Please make sure to configure it.
Username regex pattern: /(?:UPN:)([^,]+)(?:,|$)/
Access control allow origin header: https://127.0.0.1:9000
Api server url: https://127.0.0.1:9000
You will need to restart instance before your changes take full effect.
Warning: TLS is not enabled. PIV will not function until TLS is successfully configur
```

Copy

Enable multi-factor authentication login using PIV:

```
$CRIBL_HOME/bin/cribl auth mf -e true
```

The response confirms that PIV authentication is enabled and lists the current configuration. For example:

```
PIV multi-factor authentication is enabled.
Username field: san
Username regex pattern: /(?:UPN:)([^,]+)(?:,|$)/
Access control allow origin header: https://127.0.0.1:9000
Api server url: https://127.0.0.1:9000
```

Copy

Disable multi-factor authentication login using PIV:

```
$CRIBL_HOME/bin/cribl auth mf -e false
```

The response confirms that PIV authentication is disabled and lists the current configuration. For example:

```
PIV multi-factor authentication is disabled.
Username field: san
Username regex pattern: /(?:UPN:)([^,]+)(?:,|$)/
Access control allow origin header: https://127.0.0.1:9000
Api server url: https://127.0.0.1:9000
```

Copy

For a detailed example that uses the mf sub-command to configure and enable PIV authentication, see Personal Identity Verification (PIV) Authentication.

17.3.2. Boot-Start

Enables or disables Cribl Edge boot-start.

Sub-commands:

- `boot-start disable`
- `boot-start enable`

Sub-commands and Options

`boot-start disable`

Disable boot-start.

Usage:

```
./cribl boot-start disable -m initd
```

Copy

Arguments:

Option	Definition
`-m <manager>`	Init manager. Accepts: `systemd` or `initd`.
`-c <configDir>`	Config directory for the init manager.

`boot-start enable`

Enable boot-start.

Usage:

```
./cribl boot-start enable -m initd
```

Copy

Sample response:

```
Enabling Cribl to be managed by initd...
boot-start enable command needs root privileges...
Enabled Cribl to be managed by initd as user=root.
```

Copy

Arguments:

Option	Definition
`-m <manager>`	Init manager. Accepts: `systemd` or `initd`.
`-u <user>`	User to run the instance as.
`-c <configDir>`	Config directory for the init manager.

17.3.3. Cloud-Workspace

Updates an on-prem Leader Node with a new config in the `instance.yml` file called `cloudWorkspace`. With this config, the local Leader will connect to the Cribl.Cloud Leader and send it usage metrics.

Usage:

```
./cribl cloud-workspace -H <host> -p <port> # etc: flags
```

Copy

Arguments:

Option	Definition
`-d <deploymentId>`	Deployment ID used to report telemetry on multiple deployments.
`-H <host>`	Hostname/IP. Defaults to `0.0.0.0`.
`-p <port>`	Port. Defaults to `4200`.
`-n <certName>`	Name of the saved certificate to use. Mutually exclusive with `-k` or `-c`.
`-k <privKeyPath>`	Path on server to the private key to use. PEM format. Can reference `$ENV_VARS`.
`-c <certPath>`	Path on server to the certificate to use. PEM format. Can reference `$ENV_VARS`.
`-u <authToken>`	Authentication token to include as part of the connection header.
`-S <tls>`	Sets, or disables, secure communication between Leader and Worker/Edge Nodes via TLS. Accepts: ["true", "false"].
`-i <ipWhitelistRegex>`	Regex matching IP addresses that are allowed to establish a connection.
`-r <resiliency>`	Resiliency mode on the Leader. Accepts: ["none", "failover"].
`-v <failoverVolume>`	For resiliency=failover, the volume to use as shared storage.

17.3.4. Decrypt

Decrypts data with a secret key. See the related encrypt command.

Usage:

```
./cribl decrypt -v 12345 -s my-path-name
```

Copy

Arguments:

Option	Definition
`-v <value>`	Value to decrypt.
`-s <criblSecretPath>`	The cribl.secret decrypt key path. If you do not specify a cribl.secret file with -s options, the default cribl.secret is used.

17.3.5. Diag

Manages diagnostic bundles. For failover mode, see how to prevent and troubleshoot CLI problems.

Sub-commands:

- `diag create`
- `diag cpuprofile`
- `diag heapsnapshot`
- `diag list`
- `diag perf`
- `diag send`

Sub-commands and Options

`diag create`

Create a diagnostic bundle for this instance.

Usage:

```
./cribl diag create
```

Copy

Arguments:

Option	Definition
`-d`	Run create in debug mode.
`-j`	Do not append '.txt' to js files.
`-k`	Include metrics for the top 10 Sources, Destinations, Pipelines, Routes, and Packs from the last 24 hours. This matches the **Include top talkers** option in the UI and is separate from the main metrics store controlled by -M.
`-t <maxIncludeJobs>`	Maximum number of jobs (and all associated tasks) to include in the bundle.

Option	Definition
`-M`	Exclude the persistent internal metrics store from the bundle. This reduces bundle size, especially on Leaders and single-instance deployments, but does not remove other metric-related data (such as logs or data added with -k).
`-g`	Exclude git from bundle.
`-i`	Include install logs.

Windows PowerShell usage:

```
PS > $env:CRIBL_VOLUME_DIR='c:\ProgramData\Cribl'; & 'c:\Program
Files\Cribl\bin\cribl.exe' diag create
```

The Windows command requires the `CRIBL_VOLUME_DIR` environment variable. If your deployment doesn't use the default path (`c:\ProgramData\Cribl`), verify that `CRIBL_VOLUME_DIR` points to the correct directory. Otherwise, you might get empty diag files.

Sample response:

```
Created diagnostic bundle at c:\ProgramData\Cribl\diag\edge-DESKTOP-C9BEMQ2-202409251
```

Copy

Windows CMD usage:

```
> set "CRIBL_VOLUME_DIR=c:\ProgramData\Cribl" && "c:\Program
Files\Cribl\bin\cribl.exe" diag create
```

Sample response:

```
Created diagnostic bundle at c:\ProgramData\Cribl\diag\edge-DESKTOP-C9BEMQ2-202409251
```

Copy

Diagnostic Bundles and Failover

If Cribl is configured for failover, the Leader stores its runtime state (including jobs, job history, and similar data) in a special failover volume instead of the default install `/state` directory. The running Leader process sets the `CRIBL_STATE_DIR` environment variable to the failover volume, such as `<failover-`

volume>/state. However, the CLI is a separate process and does not inherit the CRIBL_STATE_DIR environment variable from the Leader.

To collect a complete diagnostic bundle, set CRIBL_STATE_DIR when you run the diag create subcommand as shown in the following example. Replace <failover-volume> with the root path of the failover volume in your environment (such as /mnt/cribl-failover) so that CRIBL_STATE_DIR resolves to the same path that the Leader uses.

```
CRIBL_STATE_DIR=<failover-volume>/state ./cribl diag create -g
```

Copy

Align the environment variables CRIBL_CONF_DIR and CRIBL_VOLUME_DIR with your deployment as for other CLI commands.

If you do not set the CRIBL_STATE_DIR environment variable in the shell, the CLI may default to <configuration-root>/state and the bundle may omit the failover state (for example, it may have empty or incomplete job data).

diag cpuprofile

Collect a 30-second CPU profile and place in the diag directory.

Usage:

```
./cribl diag cpuprofile -p 12345
```

Copy

Sample response:

```
Created a Cribl diagnostic bundle at /opt/cribl/diag/<product>-zedborcdb72f-20210820T
```

Copy

Arguments:

Option	Definition
-p <pid>	The pid of the process to dump the CPU profile.

`diag heapsnapshot`

Generate heap snapshot of a Cribl process and place in the `diag` directory.

Usage:

```
./cribl diag heapsnapshot -p 12345
```

Copy

Sample response:

```
Heap-1672574400000-12345.heapsnapshot
```

Copy

The response format is `Heap-<epoch-timestamp>-<pid>.heapsnapshot`.

Arguments:

Option	Definition
`-p <pid>`	The pid of the process to dump the heap snapshot.

`diag list`

List existing diagnostic bundles.

`diag perf`

Collect a 30-second CPU profile and a heapsnapshot and place both in the `diag` directory.

Option	Definition
`-p <pid>`	The pid of the process to dump both the CPU profile and heap snapshot.

`diag send`

Send diagnostics bundle to Cribl Support.

```
./cribl diag send -c 00001234
```

Copy

Arguments:

Option	Definition
`-c <caseNumber>`	Cribl Support Case Number.
`-p <path>`	Diagnostic bundle path. If empty, it creates a new bundle.

17.3.6. Encrypt

Encrypts data with a secret key. See the related decrypt command.

Usage:

```
./cribl encrypt -v my_token -s my-path-name
```

Copy

Sample response:

```
LzmJ3uqA6WXtXjaLPwuM++WX84o3fPxBaubA6Q0look=
```

Copy

Arguments:

Option	Definition
`-v <value>`	Value to encrypt.
`-s <criblSecretPath>`	The cribl.secret encrypt key path. If you do not specify a cribl.secret file with -s options, the default cribl.secret is used.

17.3.7. Git

Manages Fleet configuration.

Sub-commands:

- `git commit`
- `git commit-deploy`
- `git deploy`
- `git list-groups`

Sub-commands and Options

`git commit`

Commit Worker Group or Fleet configuration changes.

Usage:

```
./cribl git commit -g production_logs -m "Add syslog pipeline"
./cribl git commit -g webserver_data -m "Remove Parser function from webhook pipeline
```

Copy

Arguments:

Option	Definition
`-g <group>`	Group ID.
`-m <message>`	Commit message.

`git commit-deploy`

Commit and also deploy Worker Group or Fleet configuration changes.

Usage:

```
./cribl git commit-deploy -g production_logs -m "Add syslog pipeline"
./cribl git commit-deploy -g webserver_data -m "Remove Parser function from webhook p
```

Copy

Option	Definition
`-g <group>`	Group ID.
`-m <message>`	Commit message.

`git deploy`

Deploy Worker Group or Fleet configuration changes.

Usage:

```
./cribl git commit-deploy -g production_logs -v 7c04de1
```

Copy

Sample response:

```
Successfully committed version 7c04de1
```

Copy

Arguments:

Option	Definition
`-g <group>`	Group ID.
`-v <version>`	Deploy version.

`git list-groups`

List Worker Groups or Fleets.

17.3.8. Help

Displays a list of commands with a description (help) for each. Defaults to a selection of generally useful commands.

Usage:

```
./cribl help [-a]
```

Copy

Sample response:

```
Software version: 4.4.2
Usage: [sub-command] [options] [args]

Commands:
help              - Display help
mode-edge         - Configure instance in Edge mode
mode-managed-edge - Configure instance in Managed-Edge mode
mode-master       - Configure instance in Leader mode
mode-single       - Configure instance in Single-Instance mode
mode-worker       - Configure instance in Worker mode
reload            - Reload instance
restart           - Restart instance
start             - Start instance
status            - Display status
stop              - Stop instance
version           - Print version
```

Copy

Arguments:

Option	Definition
`-a`	Display the list of all commands, except for `scope`.

17.3.9. Keys

Manages encryption keys.



You must append the `-g <group>` argument to specify a Worker Group/Fleet. As a fallback, append the argument `-g default`. For example: `./cribl keys list -g default`.

Sub-commands:

- `keys add`
- `keys list`

Sub-commands and Options

`keys add`

Add encryption keys.

Usage:

```
./cribl keys add -g default
./cribl keys add -g production_logs
```

Copy

Sample response:

```
Adding key succeeded. Key count=1
```

Copy

Arguments:

Option	Definition
`-a <algorithm>`	Encryption algorithm. Supported values: aes-256-cbc (default), aes-256-gcm.

Option	Definition
`-c <keyclass>`	Key class to set for the key.
`-k <kms>`	KMS to use. Must be configured; see cribl.yml.
`-e <expires>`	Expiration time, in Unix time.
`-i`	Use an initialization vector. (IV size configurable for algorithm aes-256-gcm & fixed to 16 for algorithm aes-256-cbc)
`-s <ivSize>`	(For algo 'aes-256-gcm' only) Initialization vector size (bytes). Supported values: 12 (default), 13, 14, 15, 16
`-g <group>`	Worker Group ID.

`keys list`

List encryption keys.

Usage:

```
./cribl keys add -g default
./cribl keys add -g production_logs
```

Copy

Arguments:

Option	Definition
`-g <group>`	Worker Group ID.

17.3.10. Limits

Controls the availability of Cribl features. See Scripts for more information.

Usage:

```
./cribl limits -s true
./cribl limits -s false
```

Copy

Arguments:

Option	Definition
`-s <enable>`	Enable custom scripts API. Accepts: [“true”, “false”].

17.3.11. Mode-Master

Configures instance as a Leader.

For more information, see:

- Leader High Availability/Failover
- Set Up Leader and Worker Nodes

Usage:

```
./cribl mode-master -r failover -v /tmp/shared
```

Copy

Arguments:

Option	Definition
`-d <deploymentId>`	Deployment ID for reporting telemetry on multiple deployments.
`-H <host>`	Hostname/IP. Defaults to `0.0.0.0`.
`-p <port>`	Port. Defaults to `4200`.
`-f <uiAccess>`	Controls whether the distributed port responds to requests on behalf of the API process. The default port is `4200`. When disabled, `4200` only responds to Worker/Edge Node queries for config bundles and distributed upgrades. Accepts: ["true", "false"]. Defaults to `true` for backward compatibility.
`-n <certName>`	Name of the saved certificate to use. Mutually exclusive with -k or -c.
`-k <privKeyPath>`	Path on server to the private key to use. Accepts PEM format. Can reference `$ENV_VARS`.
`-c <certPath>`	Path on server to the certificate to use. Accepts PEM format. Can reference `$ENV_VARS`.
`-u <authToken>`	Authentication token to include as part of the connection header.
`-S <tls>`	Sets, or disables, secure communication between Leader and Worker/Edge Nodes via TLS. Accepts: ["true", "false"].

Option	Definition
`-i <ipWhitelistRegex>`	Regex matching IP addresses that are allowed to establish a connection.
`-r <resiliency>`	Resiliency mode for the Leader. Accepts: [“none”, “failover”].
`-v <failoverVolume>`	For resiliency=failover, the volume to use as shared storage.

17.3.12. Mode-Single

Configures node as a Single-instance Cribl Stream deployment.

Usage:

```
./cribl mode-single -d d0e301e4-f961-4f50-956a-5153224b34e3
```

Copy

Arguments:

Option	Definition
`-d <deploymentId>`	Deployment ID for reporting telemetry on multiple deployments.

17.3.13. Mode-Edge

Configures node as a Single-instance Cribl Edge deployment.

Usage:

```
./cribl mode-edge -d d0e301e4-f961-4f50-956a-5153224b34e3
```

Copy

Arguments:

Option	Definition
`-d <deploymentId>`	Deployment ID for reporting telemetry on multiple deployments.
`-H <host>`	Hostname/IP. Defaults to `127.0.0.1`.
`-p <port>`	Port. Defaults to `9420`.

17.3.14. Mode-Worker

Configures instance as a Cribl Stream Worker Node.

Running this command resets all the related options. This means that even if you want to change only one option, you also need to provide values for all the other arguments.

When you generate bootstrap scripts through the UI to add or update Workers, these scripts automatically set the `-S` option according to the Leader TLS configuration.

For more information, see Set Up Leader and Worker Nodes.

Usage:

```
./cribl mode-worker -H 127.0.0.1 -p 9420 -u myToken
```

Copy

The `-H <host> -p <port> -u <authToken>` parameters are required for successful connection to the Leader.

Arguments:

Option	Definition
`-d <deploymentId>`	Deployment ID for reporting telemetry on multiple deployments.
`-H <host>`	Leader's hostname or IP address. Required.
`-p <port>`	Leader's cluster communications port. Example: `4200`. Required.
`-n <certName>`	Name of the saved certificate to use. Mutually exclusive with `-k` or `-c`.
`-k <privKeyPath>`	Path on server to the private key to use. Accepts PEM format. Can reference `$ENV_VARS`.
`-c <certPath>`	Path on server to the certificate to use. Accepts PEM format. Can reference `$ENV_VARS`.
`-u <authToken>`	Authentication token to include as part of the connection header. Required.

Option	Definition
`-S <tls>`	Sets, or disables, secure communication between Leader and Worker/Edge Nodes via TLS. Accepts: [“true”, “false”].
`-e <envRegex>`	Regex to select environment variables to report to Leader. For example: `"/test/"`.
`-t <tags>`	Tag values to report to Leader.
`-g <group>`	Worker Group to report to Leader.

In Cribl.Cloud, get the token for your Workspace from the Edge Node modal (**Add/Update Edge Node**) from the **Auth token** field.

In an on-prem deployment, get the token from **Settings** > **Global** > **System** > **Distributed Settings** > **Leader Settings** > **Auth token**.

17.3.15. Mode-Managed-Edge

Configures instance as a Cribl Edge Node.

Running this command resets all the related options. This means that even if you want to change only one option, you also need to provide values for all the other arguments.

When you generate bootstrap scripts via the UI to add or update Workers, these scripts automatically set the `-S` option according to the Leader TLS configuration.

Usage:

```
./cribl mode-managed-edge 127.0.0.1 -p 9420 -u myToken && ./cribl restart
```

Copy

The `-H <host> -p <port> -u <authToken>` parameters are required for successful connection to the Leader.

Arguments:

Option	Definition
`-d <deploymentId>`	Deployment ID for reporting telemetry on multiple deployments.
`-H <host>`	Leader's hostname or IP address. Required.
`-p <port>`	Leader's cluster communications port Example: `4200`. Required.
`-n <certName>`	Name of the saved certificate to use. Mutually exclusive with `-k` or `-c`.
`-k <privKeyPath>`	Path on server to the private key to use. Accepts PEM format. Can reference `$ENV_VARS`.
`-c <certPath>`	Path on server to the certificate to use. Accepts PEM format. Can reference `$ENV_VARS`.
`-u <authToken>`	Authentication token to include as part of the connection header. Required.
`-S <tls>`	Sets, or disables, secure communication between Leader and Worker/Edge Nodes via TLS. Accepts: ["true", "false"].

Option	Definition
`-e <envRegex>`	Regex to select environment variables to report to Leader. Example: `"/test/"`.
`-t <tags>`	Tag values to report to Leader.
`-g <group>`	Fleet to report to Leader.

In Cribl.Cloud, get the token for your Workspace from the Edge Node modal (**Add/Update Edge Node**) from the **Auth token** field.

In an on-prem deployment, get the token from **Settings** > **Global** > **System** > **Distributed Settings** > **Leader Settings** > **Auth token**.

17.3.16. Mode-Outpost

Configures instance as a Cribl Outpost.

For more information, see:

- Cribl Outpost
- Set up Outpost

Usage:

```
./cribl mode-outpost \
  -H leader.example.com \
  -p 4200 \
  -u myToken \
  -S 1
```

Copy

The `-H <host> -p <port> -u <authToken>` parameters are required for successful connection to the Leader.

Arguments:

Option	Definition
`-d <deploymentId>`	Deployment ID for reporting telemetry on multiple deployments.
`-H <host>`	Leader's hostname or IP address.
`-p <port>`	Leader's cluster communications port.
`-n <certName>`	Name of the saved certificate to use. Mutually exclusive with -k or -c.
`-k <privKeyPath>`	Path on server to the private key to use. Accepts PEM format. Can reference `$ENV_VARS`.
`-c <certPath>`	Path on server to the certificate to use. Accepts PEM format. Can reference `$ENV_VARS`.
`-u <authToken>`	Authentication token to include as part of the connection header. Required.

Option	Definition
`-S <tls>`	Sets, or disables, secure communication between Leader and the Outpost via TLS. Accepts: [“true”, “false”].
`-e <envRegex>`	Regex to select environment variables to report to Leader. Example: `"/test/"`.
`-t <tags>`	Tag values to report to Leader.

In Cribl.Cloud, get the token for your Workspace from the Edge Node modal (**Add/Update Edge Node**) from the **Auth token** field.

In an on-prem deployment, get the token from **Settings** > **Global** > **System** > **Distributed Settings** > **Leader Settings** > **Auth token**.

17.3.17. Nc

Listens on a port for traffic, and outputs stats and data. (Netcat-like utility.)

Usage:

```
./cribl nc -p 4200
```

Copy

Sample response:

```
2021-08-20T22:44:30.457Z - starting server on 0.0.0.0:9999
2021-08-20T22:44:30.462Z - server listening 0.0.0.0:9999
2021-08-20T22:44:31.461Z - messages: 0, socks: 0, thruput: 0MBps
2021-08-20T22:44:32.466Z - messages: 0, socks: 0, thruput: 0MBps
...
2021-08-20T22:44:39.212Z - got connection: 127.0.0.1:37190
2021-08-20T22:44:39.213Z - got connection: 127.0.0.1:37192
```

Copy

Arguments:

Option	Definition
`-p <port>`	Port to listen on.
`-f <family>`	If this is `6` then nc will listen on `::1`. Otherwise, it listens on `0.0.0.0`.
`-s <statsInterval>`	Stats output interval (ms), use `0` to disable.
`-u`	Listen on UDP port instead.
`-o`	Output received data to stdout.
`-t <throttle>`	Throttle rate in (unit)/sec, where units can be KB, MB, GB, and TB.

17.3.18. Node

When you run `node` with no options, it displays a command prompt, as shown here:

```
>
```

Copy

To execute a JavaScript file, you can enter path/file name at the prompt.

With the `-v` option, prints the version of NodeJS that is running.

With `-e`, evaluates a string. Write to console to see the output. For example:

Usage:

```
./cribl node -e 'console.log(Date.now())'
./cribl node -v
```

Copy

Sample response:

```
./cribl node -e 'console.log(Date.now())'
1629740667695
```

Copy

```
v14.15.1
```

Copy

Arguments:

Option	Definition
`-e <eval>`	String to evaluate.
`-v`	Prints NodeJS version.

17.3.19. Pack

Manages Cribl Packs.

Sub-commands:

- `pack export`
- `pack install`
- `pack list`
- `pack uninstall`
- `pack upgrade`

Sub-commands and Options

`pack export`

Exports Cribl packs.

Usage:

```
./cribl pack export -m merge
```

Copy

Arguments:

Option	Definition
`-m <mode>`	Mode to export. Accepts: merge_safe, merge, default_only.
`-o <filename>`	Where to export the Pack on disk.
`-n <name>`	Name to override the installed Pack's name on export.
`-g <group>`	The Worker Group/Fleet to execute within.

`pack install`

Install a Cribl Pack.

Usage:

```
./cribl pack install -g default https://packs.cribl.io/dl/cribl-sysdig-events/1.0.3/c
```

Copy

In this command, `default` Worker Group ID.

Sample response:

```
{"time":"2025-11-13T08:14:05.599Z","cid":"api","channel":"ConfFeatureFlagsMgr","level
{"time":"2025-11-13T08:14:05.600Z","cid":"api","channel":"ConfFeatureFlagsMgr","level
{"time":"2025-11-13T08:14:05.661Z","cid":"api","channel":"collectors","level":"info",
{"time":"2025-11-13T08:14:05.671Z","cid":"api","channel":"LookupDiskUsageLimiterFacto
{"time":"2025-11-13T08:14:05.672Z","cid":"api","channel":"LookupDiskUsageLimiterFacto
{"time":"2025-11-13T08:14:05.687Z","cid":"api","channel":"collectors","level":"info",
{"time":"2025-11-13T08:14:05.691Z","cid":"api","channel":"PackMgr","level":"info","me
{"time":"2025-11-13T08:14:05.695Z","cid":"api","channel":"PackMgr","level":"info","me
{"time":"2025-11-13T08:14:05.695Z","cid":"api","channel":"PackMgr","level":"info","me
{"time":"2025-11-13T08:14:06.330Z","cid":"api","channel":"PackMgr","level":"info","me
https://packs.cribl.io/dl/cribl-sysdig-events/1.0.3/cribl-sysdig-events-1.0.3.crbl: s
{"time":"2025-11-13T08:14:06.993Z","cid":"api","channel":"PackMgr","level":"info","me
```

Copy

Arguments:

Option	Definition
`-d`	Run install in debug.
`-f`	Force install.
`-c`	Disallow installation of Packs with custom functions.
`-n <name>`	Name of the Pack to install, defaults to source.
`-g <group>`	The Worker Group/Fleet to execute within. Required for distributed deployments.

pack list

Lists Cribl Packs.

Usage:

```
./cribl pack list
```

Copy

Arguments:

Option	Definition
`-v`	Display all Pack information in verbose mode.
`-g <group>`	The Worker Group/Fleet to execute within.

pack uninstall

Uninstall a Cribl Pack.

Usage:

```
./cribl pack uninstall
```

Copy

Arguments:

Option	Definition
`-d`	Run uninstall in debug.
`-g <group>`	The Worker Group/Fleet to execute within.

pack upgrade

Upgrade a Cribl Pack.

Usage:

```
./cribl pack upgrade -s https://packs.cribl.io/packs/cribl-sysdig-events
```

Copy

Arguments:

Option	Definition
`-d`	Run upgrade in debug.
`-s <source>`	Provide the Pack source.
`-m <minor>`	Only upgrade to minor version.
`-g <group>`	The Worker Group/Fleet to execute within.

17.3.20. Parquet

For any Parquet file: view (`cat`) the file, its metadata (`details`), or its `schema`.



This command is available only on Linux.

Sub-commands:

- `parquet cat`
- `parquet details`
- `parquet schema`

Sub-commands and Options

`parquet cat`

Cat Parquet events to console.

Usage:

To view a Parquet file:

```
./cribl parquet cat -f /some_parquet_file
```

Copy

Sample response:

```
{"__bytes":274,"Unnamed: 0":0,"name":"apples","quantity":10,"price":2.6,"date":"2018-
...
```

Copy

Arguments:

Option	Definition
`-f <file>`	Path to Parquet file.
`-m <max>`	Maximum events to print.

`parquet details`

View all Parquet details (schema and statistics) and metadata, including encoding and compression.

Usage:

```
./cribl parquet details -f /some_parquet_file
```

Copy

Sample response:

```
{
  "schema": {
    "Unnamed: 0": {
      "optional": true,
      "type": "INT64"
    },
    ...
  },
  "path": "../../../src/sluice/js/util/__tests__/data/parquet/single/small_fruits.par
  "num_rows": 40000,
  "num_cols": 11,
  "num_real_cols": 11,
  "num_row_groups": 1,
  "created_by": "parquet-cpp-arrow version 11.0.0",
  "schema_root_name": "schema",
  "metadata": {
    "pandas": "{\"index_columns\": [{\"kind\": \"range\", \"name\": null, \"start\":
    ...
  },
  "version": "PARQUET_2_6",
  "metadataSize": 6072,
  ...
    }
  ]
}
```


Copy

Arguments:

Option	Definition
`-f <file>`	Path to Parquet file.

`parquet schema`

View a Parquet file's schema (not including encoding or compression), expressed as JSON like in the Parquet Schema editor.

Usage:

```
./cribl parquet schema -f /some_parquet_file
```

Copy

Sample response:

```
{
  "Unnamed: 0": {
    "optional": true,
```

17.3.21. Pipe

Feeds standard input (stdin) to a Pipeline.

Usage:

```
cat sample.log |  ./cribl pipe -p pipelineName
cat sample.log |  ./cribl pipe -p pipelineName 2>/dev/null
```

Copy

Sample response:

```
...
{"time":"2021-08-20T20:37:00.017Z","cid":"api","channel":"commands","level":"info","m
{"time":"2021-08-20T20:37:00.019Z","cid":"api","channel":"pipe:main","level":"info","
{"time":"2021-08-20T20:37:00.021Z","cid":"api","channel":"pipe:main","level":"info","
{"time":"2021-08-20T20:37:00.022Z","cid":"api","channel":"commands","level":"info","m
{"time":"2021-08-20T20:37:00.028Z","cid":"api","channel":"GrokMgr","level":"info","me
...
```

Copy

Arguments:

Option	Definition
`-p <pipeline>`	Pipeline to feed data through.
`-d`	Include dropped events.
`-c <cpuProfile>`	Perform CPU profiling.
`-t`	Perform pipeline tracing.
`-a <pack>`	Optional Cribl Pack context. Mutually exclusive with -b.
`-b <project>`	Optional Cribl Project context.
`-i`	Runs the pipe command without sandboxing javascript expressions from potential attackers.

17.3.22. Pq

Benchmark persistent queue (PQ) performance. Use `pq write` to simulate high-throughput event ingestion and `pq read` to simulate PQ consumption. This command is useful for internal benchmarking and evaluating disk performance in customer environments.

Sub-commands:

- `pq read`
- `pq write`

Sub-commands and Options

`pq-read`

Read data from a persistent queue and discard it. Use this command to test PQ deserialization and consumer performance.

Usage:

```
./cribl pq read -o /tmp/pq
```

Copy

Sample response:

This response prints every second during operation:

```
{"time":"2025-07-15T22:38:37.978Z","cid":"api","channel":"IPersistentQueue","level":'
{"time":"2025-07-15T22:38:37.982Z","cid":"api","channel":"IPersistentQueue","level":'
{"time":"2025-07-15T22:38:38.978Z","cid":"api","channel":"ProcessMetrics","level":"ir
{"time":"2025-07-15T22:38:38.982Z","cid":"api","channel":"commands:pq","level":"info'
{"time":"2025-07-15T22:38:39.978Z","cid":"api","channel":"ProcessMetrics","level":"ir
{"time":"2025-07-15T22:38:39.982Z","cid":"api","channel":"commands:pq","level":"info'
{"time":"2025-07-15T22:38:40.978Z","cid":"api","channel":"ProcessMetrics","level":"ir
{"time":"2025-07-15T22:38:40.982Z","cid":"api","channel":"commands:pq","level":"info'
{"time":"2025-07-15T22:38:41.978Z","cid":"api","channel":"ProcessMetrics","level":"ir
{"time":"2025-07-15T22:38:41.982Z","cid":"api","channel":"commands:pq","level":"info'
{"time":"2025-07-15T22:38:42.978Z","cid":"api","channel":"ProcessMetrics","level":"ir
{"time":"2025-07-15T22:38:42.982Z","cid":"api","channel":"commands:pq","level":"info'
{"time":"2025-07-15T22:38:43.978Z","cid":"api","channel":"ProcessMetrics","level":"ir
{"time":"2025-07-15T22:38:43.982Z","cid":"api","channel":"commands:pq","level":"info'
{"time":"2025-07-15T22:38:44.978Z","cid":"api","channel":"ProcessMetrics","level":"ir
{"time":"2025-07-15T22:38:44.982Z","cid":"api","channel":"commands:pq","level":"info'
{"time":"2025-07-15T22:38:45.978Z","cid":"api","channel":"ProcessMetrics","level":"ir
{"time":"2025-07-15T22:38:45.982Z","cid":"api","channel":"commands:pq","level":"info'
{"time":"2025-07-15T22:38:46.978Z","cid":"api","channel":"ProcessMetrics","level":"ir
{"time":"2025-07-15T22:38:46.982Z","cid":"api","channel":"commands:pq","level":"info'
{"time":"2025-07-15T22:38:47.146Z","cid":"api","channel":"IPersistentQueue","level":'
{"time":"2025-07-15T22:38:47.148Z","cid":"api","channel":"commands:pq","level":"info'
```


Copy

Arguments:

Option	Definition
`-o <path>`	**(Required)** Directory path to read persistent queue files from.

This command does not commit or remove files from the persistent queue. It's intended for read-only performance testing.

`pq-write`

Read data from `stdin` and write to a persistent queue. Use this command to test PQ serialization and disk write performance.

Usage:

```
cat ~/work/syslog_large.log | ./cribl pq write -o /tmp/pq -b 1000 -s 10485760 -c gzip
```

Copy

Sample response:

This response prints every second during operation:

```
{"time":"2025-07-15T22:36:45.543Z","cid":"api","channel":"IPersistentQueue","level":'
{"time":"2025-07-15T22:36:45.545Z","cid":"api","channel":"IPersistentQueue","level":'
{"time":"2025-07-15T22:36:46.543Z","cid":"api","channel":"ProcessMetrics","level":"ir
{"time":"2025-07-15T22:36:46.545Z","cid":"api","channel":"commands:pq","level":"info'
{"time":"2025-07-15T22:36:47.543Z","cid":"api","channel":"ProcessMetrics","level":"ir
{"time":"2025-07-15T22:36:47.545Z","cid":"api","channel":"commands:pq","level":"info'
{"time":"2025-07-15T22:36:48.543Z","cid":"api","channel":"ProcessMetrics","level":"ir
{"time":"2025-07-15T22:36:48.545Z","cid":"api","channel":"commands:pq","level":"info'
{"time":"2025-07-15T22:36:49.542Z","cid":"api","channel":"ProcessMetrics","level":"ir
{"time":"2025-07-15T22:36:49.545Z","cid":"api","channel":"commands:pq","level":"info'
{"time":"2025-07-15T22:36:50.543Z","cid":"api","channel":"ProcessMetrics","level":"ir
{"time":"2025-07-15T22:36:50.545Z","cid":"api","channel":"commands:pq","level":"info'
{"time":"2025-07-15T22:36:51.543Z","cid":"api","channel":"ProcessMetrics","level":"ir
{"time":"2025-07-15T22:36:51.545Z","cid":"api","channel":"commands:pq","level":"info'
{"time":"2025-07-15T22:36:52.542Z","cid":"api","channel":"ProcessMetrics","level":"ir
{"time":"2025-07-15T22:36:52.545Z","cid":"api","channel":"commands:pq","level":"info'
{"time":"2025-07-15T22:36:52.563Z","cid":"api","channel":"commands:pq","level":"info'
```

Copy

Output includes the number of events and bytes processed every second, and a final summary on completion.

Arguments:

Option	Definition
`-o <path>`	**(Required)** Directory path to store persistent queue files.

17.3.23. Reload

Reloads Cribl Edge. Executes immediately.

Usage:

```
`./cribl reload`
```

Copy

Sample response:

```
Reload request submitted to Cribl
```

Copy

17.3.24. Restart

Restarts Cribl Edge. Executes immediately.



Executing this command cancels any running collection jobs.

Usage:

```
./cribl restart
```

Copy

Sample response:

```
Stopping Cribl, process 17
..............
Cribl is not running
Starting...
....
Cribl started
```

Copy

17.3.25. Start

Starts Cribl Edge. Executes immediately. Upon first run, echoes the Cribl Edge default login credentials.

Usage:

```
./cribl start
```

Sample response:

```
Starting...
...
Cribl started
```

Copy

Arguments:

Option	Definition
`-d <dir>`	Configuration directory.
`-r <role>`	Process role.

17.3.26. Status

Displays status of Cribl Edge, including the API Server address, instance's mode (Leader or Worker), process ID, and GUID. Executes immediately.

Usage:

```
./cribl status
```

Copy

Sample response:

```
Address: http://172.17.0.3:9000
Mode: master
Status: Up
Software Version: 3.1.0-f765e418
Config Version: 347079c
Master: 0.0.0.0:4200
PID: 4100
GUID: e706052a-ace9-4511-a7c7-b58a414a07d3
```

Copy

17.3.27. Stop

Stops Cribl Edge. Executes immediately.



Executing this command cancels any running collection jobs.

Usage:

```
./cribl stop
```

Copy

Sample response:

```
Stopping Cribl, process 3951
............
Cribl is not running
```

Copy

17.3.28. Vars

Manages Cribl Edge global variables.

See Global Variables for more information.

Sub-commands:

- `vars add`
- `vars get`
- `vars remove`
- `vars update`

Sub-commands and Options

`vars add`

Add a global variable.

Usage:

```
./cribl vars add -i exampleVariable -v 42 -t number -d "Sample number variable" -c cr
```

Copy

Sample response:

```
[
  {
    "type": "number",
    "lib": "cribl",
    "description": "Sample number variable",
    "value": "42",
    "tags": "cribl,sample",
    "id": "exampleVariable"
  }
]
```

Copy

Arguments:

Option	Definition
`-i <id>`	Global variable ID.
`-t <type>`	Type.
`-v <value>`	Value.
`-a <args>`	Arguments.
`-d <description>`	Description.
`-c <tags>`	Custom Tags (comma separated list).
`-g <group>`	Group ID.

`vars get`

List global variables.

Usage:

```
./cribl vars get
```

Copy

Arguments:

Option	Definition
`-i <id>`	Global variable ID.
`-g <group>`	Group ID.

`vars remove`

Remove global variable.

Usage:

```
./cribl vars remove -i exampleVariable
```

Copy

Arguments:

Option	Definition
`-i <id>`	Global variable ID.
`-g <group>`	Group ID.

`vars update`

Update global variable.

Usage:

```
./cribl vars update -i exampleVariable -v 2001
```

Copy

Arguments:

Option	Definition
`-i <id>`	Global variable ID.
`-t <type>`	Type.
`v <value>`	Value.
`-a <args>`	Arguments.
`-d <description>`	Description.
`-c <tags>`	Custom Tags (comma separated list).
`-g <group>`	Group ID.

17.3.29. Version

Displays Cribl Edge version. Executes immediately.

Usage:

```
./cribl version
```

Copy

Sample response:

```
Software Version: 4.7.3-6f48361f
```

Copy

17.4. Environment Variables

You can use the following environment variables to configure your Cribl Edge installation.

Name	Purpose
`CRIBL_TMP_DIR`	Defines the root of a temporary directory. See Usage Notes below.
`CRIBL_VOLUME_DIR`	Sets a directory that persists modified data between different containers or ephemeral instances. When set, this environment variable overrides `CRIBL_HOME`. It also creates predefined subdirectories in the specified directory. If that directory already contains subdirectories with those names, they will be overwritten.
`CRIBL_BOOTSTRAP`	Quickstart a Cribl instance by configuring this variable.
`CRIBL_BOOTSTRAP_HOST`	Host name for connecting to the Leader Node when setting up a new Edge Node.
`CRIBL_API_HOST`	Overrides the **Host** setting.
`CRIBL_API_PORT`	Overrides the **Port** setting.
`CRIBL_EDGE`	(Cribl Edge only) When set to any value, runs this command at container start: `cribl mode-edge -H 0.0.0.0`. This launches the instance as an Edge Node, listening on a host at `0.0.0.0`.
`CRIBL_EDGE_FS_ROOT`	(Cribl Edge running in a container only) Location of the host OS filesystem when mounting in a container. Defaults to `/hostfs`.
`CRIBL_K8S_POD`	(Cribl Edge on Kubernetes only) Sets the name of the Kubernetes Pod in which Cribl Edge is deployed.
`CRIBL_K8S_TLS_REJECT_UNAUTHORIZED`	(Cribl Edge on Kubernetes only) Set to `0` to disable certificates validation when connecting to the Kubernetes APIs. When you disable this environment variable, all Kubernetes features (including Metadata, Metrics, Logs, and workload instrumentation metadata) will tolerate invalid TLS certificates (such as, expired, self-signed, and so forth) when connecting to the Kubernetes APIs.
`CRIBL_SERVICEACCOUNT_PATH`	(Cribl Edge on Kubernetes only) Path to the `ServiceAccount` to use to query the Kubernetes API.

Name	Purpose
	Defaults to `/var/run/secrets/kubernetes.io/serviceaccount`.
`CRIBL_K8S_CPU_LIMIT`	(Cribl Stream on Kubernetes only) Automatically sets CPU shares for Cribl Stream based on the `resources.limits.cpu` value, configured by default in the Helm chart.
`CRIBL_MAX_WORKERS`	(Cribl Stream on Kubernetes only) Setting a maximum amount of Worker Processes on a Kubernetes node helps to prevent overloading the node with more CPUs than it can handle.
`CRIBL_NOEXEC`	When set to `1`, prevents the Exec Source from processing any data, effectively putting it into a "disabled" state. To re-enable, remove or unset the `CRIBL_NOEXEC` variable.
`CRIBL_ENABLE_CORE_DUMP`	When set to `1` in a container, enables core dumps. This requires setting `core_pattern` at the host level. Otherwise, core dumps from a container are disabled.

Distributed Deployment Variables

You can use the following environment variables to configure your distributed Cribl Edge instance.

Name	Purpose
`CRIBL_DEPLOYMENT`	Use this variable to set a human-readable name for your on-prem Leader. Example: `linux_production`. The value you set using this variable will appear on the Connected Environments page.
`CRIBL_CLOUD_WORKSPACE_URL`	URL of the Leader Node in a Cribl.Cloud deployment. Example: `CRIBL_DIST_LEADER_URL=tls://<authToken>@leader:4200`. See Usage Notes.
`CRIBL_DIST_LEADER_URL`	URL of the Leader Node. Example: `CRIBL_DIST_LEADER_URL=tls://<authToken>@leader:4200`. See Usage Notes.
`CRIBL_DIST_MODE`	`worker` or `leader`. Defaults to `worker`, if (and only if) `CRIBL_DIST_LEADER_URL` is present.
`CRIBL_DIST_WORKER_PROXY`	Communicate to the Leader Node via a SOCKS proxy. See Usage Notes below.
`CRIBL_DIST_LEADER_BUNDLE_URL`	Specifies the location of configuration bundles. When set, Edge Nodes will attempt to retrieve their configuration bundles from the

Name	Purpose
	URL specified by this variable, rather than directly from the Leader Node.

Usage Notes

This section explains how to use certain environment variables.

`CRIBL_DIST_LEADER_URL`

Use this format:

`<tls|tcp>://<authToken>@<host>:<port>?group=defaultGroup&tag=tag1&tag=tag2&tls.<tls_settings>`

> To generate a random authentication token, leave `<authToken>` unchanged. You can define it to add your own token instead, but make sure it's secure enough.
>
> The auth token can include only alphanumeric characters or the underscore (`_`). Do not include `<`, `>`, `"`, `` ` ``, `\r`, `\n`, `\t`, `{`, `}`, `|`, `\`, `^`, or `'`.

When configuring the node as a Leader, use `0.0.0.0` (or another suitable binding address) for `host`. Then, set `CRIBL_DIST_MODE` to `leader`.

Here are the components:

- `group` - The preferred Fleet assignment.
- `resiliency` - The preferred Leader failover mode.
- `volume` - The location of the NFS directory to support Leader failover.
- `tag` - A list of tags that you can use to map (Stream, Edge) the Worker to a Fleet.
- `disableSNIRouting` - Whether Server Name Indicator (SNI) routing is enabled on the Edge Node. Boolean; defaults to `false`.

  > Do not edit this advanced setting without supervision by Cribl Support. Changing this setting can affect the scalability of your system because it determines how Cribl Edge routes connections between Leader Nodes and Edge Nodes. See the Known Issue that resulted in this setting for more details.

- `tls.privKeyPath` - Private Key Path.

- `tls.passphrase` - Key Passphrase.
- `tls.caPath` - CA Certificate Path.
- `tls.certPath` - Certificate Path.
- `tls.rejectUnauthorized` - Validate Client Certs. Boolean; defaults to `false`.
- `tls.requestCert` - Authenticate Client (mutual auth). Boolean; defaults to `false`.
- `tls.commonNameRegex` - Regex matching peer certificate > subject > common names allowed to connect. Used only if `tls.requestCert` is set to `true`.
- `compression` - Codec used for compressing the data before sending. One of: `gzip` or `none`.



`CRIBL_DIST_LEADER_URL` was previously called `CRIBL_DIST_MASTER_URL`. This name is still in use in certain contexts, for example in Helm charts.

CRIBL_TMP_DIR

Sources use this variable to construct temporary directories in which to stage downloaded Parquet data. If `CRIBL_TMP_DIR` is not set (the default), Cribl applications create subdirectories within your operating system's default temporary directory:

- For Cribl Stream: `<OS_default_temporary_directory>/stream/`.
- For Cribl Edge: `<OS_default_temporary_directory>/edge/`.

For example, on Linux, Stream's default staging directory would be `/tmp/stream/`.

If you explicitly set this `CRIBL_TMP_DIR` environment variable, its value replaces this OS-specific default parent directory.

CRIBL_DIST_WORKER_PROXY

`CRIBL_DIST_WORKER_PROXY` lets you configure a SOCKS proxy for communication between the Leader Node and Edge Nodes.

The default protocol is SOCKS5, which uses the format: `socks5://<username>:<password>@<host>:<port>`. Only `<host>:<port>` are required.

You can alternatively specify SOCKS4 as the protocol with the format: `socks4://<proxyhost>:<port>`. To authenticate on a SOCKS4 proxy with username and password, use this format: `socks4://<username>:<password>@<proxyhost>:<port>`. The `<proxyhost>` can be a hostname, IPv4, or IPv6.

To configure a SOCKS proxy via the UI, see Configure SOCKS Proxy. To configure an HTTP(S) proxy to use for your Sources and Destinations, refer to System Proxy Configuration.

CRIBL_BOOTSTRAP_HOST

`CRIBL_BOOTSTRAP_HOST` overrides the Add/Bootstrap New Worker script generator's default hostname.

For example, if you set `CRIBL_BOOTSTRAP=myhost`, then `myhost` will appear in the script modal's **Leader or Outpost hostname/IP** field, instead of the URL used in the browser.

CRIBL_BOOTSTRAP

`CRIBL_BOOTSTRAP` enables specifying a URL, an absolute disk file path, or a YAML string, in order to bootstrap a configuration to the `$CRIBL_HOME/local` directory. Cribl Edge applies this configuration only upon its first startup.

For any method, Cribl Edge expects each targeted config file to be YAML-formatted. Each file's top-level keys should be the paths to config files inside the `$CRIBL_HOME/local/...` subdirectory.

Below is an example of a bootstrap file. Its output, when Cribl Edge starts, would be to create three files inside the `$CRIBL_HOME/local/cribl` path: `inputs.yml`, `outputs.yml`, and `pipelines/route.yml`.

```
cribl/inputs.yml:
  inputs:
    <id>:
      <config>
cribl/outputs.yml:
  outputs:
    <id>:
      <config>
cribl/pipelines/route.yml:
  id: default
  groups: {}
  comments: []
  routes:
    ...
```

Copy

For details about each file's syntax, see Config Files and its child topics.

`CRIBL_GIT_REMOTE` and `CRIBL_GIT_REMOTE_URL`

These two GitOps bootstrap variables are closely related but behave very differently. Use them with the following in mind.

`CRIBL_GIT_REMOTE` (Bootstrap/Force Sync)

When `CRIBL_GIT_REMOTE` is set on a Leader at startup, it triggers a one-time forced synchronization from the Git repo, rather than simply setting the remote URL.

On Leader startup with `CRIBL_GIT_REMOTE` set:

- Treats the remote Git repo as the source of truth for configuration.
- Replaces any uncommitted local configuration in the Leader's Git working copy with the contents of the configured remote and branch. Effectively a `git reset --hard`.
- Restarts the Leader after the sync is complete.
- Removes `CRIBL_GIT_REMOTE` from the running process so that this forced sync does not recur on subsequent restarts.

This bootstrap process runs based solely on the presence of `CRIBL_GIT_REMOTE`, even if `CRIBL_GIT_OPS` is set to `none`. If the remote repo is unreachable or misconfigured, the Leader can fail to start until the environment variable is removed or corrected.

Recommended use: For one-time bootstrap or recovery scenarios, where you explicitly want to discard uncommitted local configuration in favor of a remote branch, and you have a backup of `$CRIBL_HOME` before enabling this variable.

`CRIBL_GIT_REMOTE_URL` (Remote URL Configuration)

`CRIBL_GIT_REMOTE_URL` exists to configure the Git remote in a way that matches the Git Settings UI without triggering a forced synchronization or restart.

- It sets the Git remote URL used by the Leader.
- It does not wipe uncommitted local configuration.
- When both variables are set, `CRIBL_GIT_REMOTE_URL` takes precedence for the URL value, while `CRIBL_GIT_REMOTE` acts as the switch to enable the forced bootstrap behavior.

Recommended use: For standard Git integration where you want to maintain local changes until you explicitly commit and push them. This is best for keeping the UI's remote URL in sync via environment variables without risking data loss.

Adding Fallback Leaders

You can configure fallback Leader Nodes to support high availability and failover. Use the following environment variables.

Name	Purpose
`CRIBL_DIST_MASTER_RESILIENCY=failover`	Sets the Leader's `Resiliency` to `Failover` mo
`CRIBL_DIST_MASTER_FAILOVER_VOLUME=/tmp/shared`	Sets the location of the NFS directory to support failover. For example: `/mnt/cribl`.
`CRIBL_DIST_MASTER_FAILOVER_MISSED_HB_LIMIT`	Determines how many Lease refresh periods ca before the standby Leader Nodes attempt to pro themselves to primary. Cribl recommends settin
`CRIBL_DIST_MASTER_FAILOVER_PERIOD`	Determines how often the primary Leader refres on the Lease file. Cribl recommends setting this
`CRIBL_INSTANCE_HOME`	In `Failover` mode, this variable points to the L `root` directory, as opposed to the shared volum to access `$CRIBL_INSTANCE_HOME/local/_system/in` (`C:\ProgramData\Cribl\local\_system.ym` Edge on Windows).

GitOps Variables

Cribl Edge provides the following environment variables to facilitate using GitOps in Cribl Stream.

You can use these variables to override the defaults in the UI if you are not using systemd to manage the `cribl` service.

Bootstrap Variables

Name	Purpose
`CRIBL_GIT_REMOTE`	Location of the remote repo to track. Can contain username and password for HTTPS auth. See Usage Notes.
`CRIBL_GIT_REMOTE_URL`	Configure simple Git remote URL. Takes precedence over `CRIBL_GIT_REMOTE` for URL setting. See Usage Notes.

Name	Purpose
`GIT_SSH` / `GIT_SSH_COMMAND`	See Git's documentation.
`CRIBL_GIT_BRANCH`	Git ref (branch, tag, commit) to track/check out.
`CRIBL_GIT_AUTH`	One of: `none`, `basic`, or `ssh`.
`CRIBL_GIT_USER`	Used for `basic` auth.
`CRIBL_GIT_PASSWORD`	Used for `basic` auth.
`CRIBL_GIT_OPS`	Controls which GitOps workflow to use. One of: `push` to enable the GitOps push workflow, or `none` to disable GitOps.
`CRIBL_INTERACTIVE`	Controls whether git commands called by Cribl CLI at startup are interactive.
`CRIBL_GIT_SSH_KEY`	Content of the SSH key used to access git remote.
`CRIBL_GIT_STRICT_HOST_KEY_CHECKING`	Controls whether to check the host key strictly.

Additional Environment Variables

Cribl Edge uses the following variables for multiple purposes.

Name	Purpose
`CRIBL_SPOOL_DIR`	Specifies the base path where events from various Sources and Destinations are spooled. Defaults to `$CRIBL_HOME/state/spool` or `$CRIBL_VOLUME_DIR/state/spool`.

Cribl.Cloud Certificate Variables

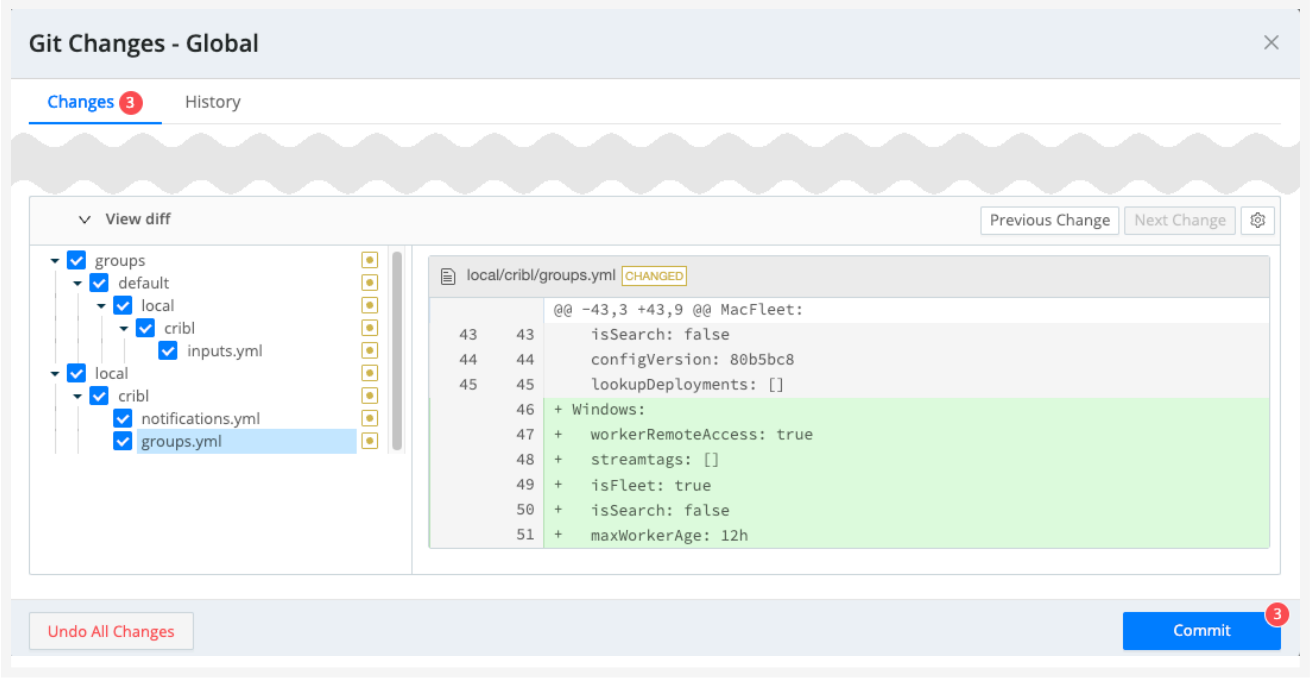
The following variables point to the location of default certificate files on Cribl.Cloud:

Name	Purpose
`CRIBL_CLOUD_KEY`	Path to the default private key on Cribl.Cloud.
`CRIBL_CLOUD_CRT`	Path to the default certificate on Cribl.Cloud.

17.5. Configuration Files

Configuration files store the configuration changes locally on your system. They reflect the changes you make to configuration settings in the UI.

You can preview how settings are translated into configuration changes in the Git Changes modal whenever you commit your changes via the Version Control option.



Git Changes screen showing Git diff with changes to configuration files.

Typically, you do not modify configuration files manually. If you need to edit the configuration outside the UI, Cribl recommends using the API. However, if your particular use case requires it, in an on-prem deployment the files are open for editing.

Configuration File Locations

In an on-prem deployment, configuration files are stored in the `$CRIBL_HOME/local/cribl` directory (where `$CRIBL_HOME` is your installation directory, for example, `/opt/cribl/`). Configuration for specific Worker Groups or Fleets is located `$CRIBL_HOME/groups/<group-or-fleet-name>/local/cribl`.

You can view the default configurations by taking a look in the `default` directory, instead of `local`.



Pack configuration resides within a separate subdirectory for each Pack: `$CRIBL_HOME/default/<pack_name>`.

Configuration file/folder	Description
`ai.yml`	Cribl AI settings.
`app-limits.yml`	Internal limit configuration.
`appscope.yml`	Deprecated legacy instrumentation configuration file.
`breakers-search.yml`	Cribl Search Event Breaker settings.
`breakers.yml`	Event Breakers configuration.
`certificates.yml`	TLS/SSL certificates.
`collectors`	Directory storing individual Collector configurations.
`conditions`	Directory storing configurations for Notification conditions.
`cribl.yml`	General system configuration.
`dataset-providers.yml`	Cribl Search Dataset Providers.
`datasets.yml`	Cribl Search Datasets.
`email-templates.yml`	Templates for Email Notifications.
`executors`	Executor job configurations.
`fleet-mappings.yml`	Mapping rulesets for Edge Nodes.
`functions`	Directory storing individual Function configurations.
`grok-patterns`	Directory storing individual Grok Pattern configurations.
`groups.yml`	Worker Group and Edge Fleet settings.
`inputs.yml`	Sources configuration.
`instance.yml`	Configuration for the current instance.
`iometrics.yml`	Metrics levels for individual Sources and Destinations in Cribl Stream.
`ipfix-information-elements.yml`	Predefined IPFIX fields for handling and decoding IPFIX fields from the Cribl Stream NetFlow & IPFIX Source.
`jobs.yml`	Collector configuration.
`job-limits.yml`	Parameters for Collection jobs and system tasks.

Configuration file/folder	Description
`kms.yml`	KMS provider configuration.
`lake-config.yml`	Configuration for Cribl Lake.
`lakes.yml`	Configuration for individual Cribl Lake datasets.
`leader.yml`	Secondary Leader configuration.
`licenses.yml`	Licenses.
`limits.yml`	Limit configuration.
`logger.yml`	Logging levels and redactions.
`mappings.yml`	Mapping rulesets for Stream Workers.
`messages.yml`	Messages displayed in the UI's `Messages` fly-out.
`mdt-devices.yml`	Configuration for MDT (Model Driven Telemetry) devices.
`notifications.yml`	Notifications and Notification targets.
`notification-templates.yml`	Templates for email Notifications.
`outpost.yml`	Configuration for Cribl Outpost.
`outputs.yml`	Destinations configuration.
`parquet-schemas.yml`	Parquet Schemas.
`parquet-schemas`	Directory storing individual Parquet Schema configurations.
`parsers.yml`	Parsers Library.
`perms.yml`	Permission configuration.
`persistent-queue.yml`	Persistent queue configuration.
`pipelines`	Directory storing individual Pipeline configurations.
`policies.yml`	RBAC Policies.
`protobuf-libraries.yml`	Configuration for Protobuf libraries.
`redis-cache-limits.yml`	Redis cache settings.

Configuration file/folder	Description
`redis-limits.yml`	Redis connection settings.
`regexes.yml`	Regex Library.
`roles.yml`	RBAC Roles.
`route.yml`	Route configuration. Located in `local/cribl/pipelines/routes.yml`.
`samples.yml`	Metadata about about stored sample data files.
`saved-queries.yml`	Saved searches in Cribl Search.
`schemas.yml`	Schema Library.
`schemas`	Directory storing individual schema configurations.
`scripts.yml`	Scripts configured at **Settings** > **Global** > **Scripts**.
`sds-rules.yml`	Cribl Guard rules.
`sds-rulesets.yml`	Cribl Guard rulesets.
`search-limits.yml`	Limit configuration for Cribl Search.
`search-usage-groups`	Cribl Search Usage Group configuration.
`secrets.yml`	Secrets.
`scope_protocol.yml`	Deprecated legacy instrumentation configuration for protocol detection.
`scope.yml`	Deprecated legacy instrumentation configuration.
`search-limits.yml`	Cribl Search limits.
`search-usage-groups.yml`	Cribl Search Usage Groups.
`service.yml`	Service processes.
`vars.yml`	Variables Library.

Configurations and Restart



You can **Restart** and **Reload** via the UI. In the sidebar, select **Settings**, then **Global**. Under **System** >

> **Controls** select **Reload** or **Restart**.

In a distributed environment, Edge Nodes poll the Leader for configuration changes. Many of these changes require a quick reload to read the new configuration, while others require a restart of the Cribl processes on the Edge Node.

Upon restarts, be aware of the following:

- Syslog data still being received over UDP might be dropped.
- Edge Nodes will temporarily disappear from the Leader's **Workers** or **Edge Nodes** page.
- Aggregation and suppression operations will start over.
- Edge Nodes' local copies of Monitoring metrics will be dropped.
- Cribl Edge will drop any events still in RAM that were bound for persistent queues. (However, PQ data already written to disk will persist through the restart.)

Changes that require reloads include configuration changes to:

- Functions
- Pipelines
- Packs
- Routes
- Lookups
- Parquet schemas
- Global variables
- Fleet Settings > Limits
- Fleet Settings > Logging > Levels

Changes that require restarts include configuration changes to:

- Distributed mode (Leader versus Managed Edge Node or Single instance)
- Fleet assignment
- Event Breakers
- QuickConnect configs
- Sources
- Destinations
- Fleet Settings > General Settings > TLS
- Fleet Settings > General Settings > Advanced

- Fleet Settings > Worker Processes > Process count and Memory

Some general guidelines to keep in mind:

- Configuration changes generated by most UI interactions - for instance, changing the order of Functions in a Pipeline, or changing the order of Routes - **do not require restarts**.
- Some configuration changes in the **Settings** UI **do require restarts**. These will prompt you for confirmation before restarting.
- All direct edits of configuration files in `$CRIBL_HOME/local/` **will require restarts**.
- Edge Nodes might temporarily disappear from the Leader's **Workers** or **Edge Nodes** tab while restarting.
- A `git commit` command on the Leader Node's host (using a freestanding `git` client not embedded in Cribl's CLI or UI) **will require either a reload or restart**.
- When using the Cribl App for Splunk, changes to Splunk configuration files might or might not require

17.5.1. Breakers.Yml

Event Breaker configuration is stored in `breakers.yml`.

breakers.yml

```
# ID - ID can contain only letters (A-Z, a-z), numbers (0-9), dashes, underscores, ar
# [string; required]
id:
# Library - Library classification for the ruleset
# One of: custom | cribl-custom
# [string; default: custom]
lib:
# Description - Description of the event breaker ruleset
# [string]
description:
# Tags - Tags associated with the ruleset
# [string]
tags:
# Min raw length - The minimum number of characters in _raw to determine which rule t
# [number; min: 50; max: 100000; default: 256]
minRawLength:
# Rules - A list of rules that will be applied, in order, to the input data stream
rules:
  # Name - Name of the rule
  # [string; required]
  name:
  # Filter condition - JavaScript expression applied to the beginning of a file or ob
  # whether the rule applies to all contained events.
  # [string; required]
  condition:
  # Event Breaker type - Type of event breaker to use
  # One of: regex | json | json_array | header | timestamp | csv | aws_cloudtrail | a
  # [string; required]
  type:
  # Index - Rule index for ordering
  # [number]
  index:
  # Timestamp anchor - The regex to match before attempting timestamp extraction. Use
  # anchor) to prevent extraction.
  # [string; required]
  timestampAnchorRegex:
  # Default timezone - Default timezone for timestamp parsing
  # [string; required]
  timestampTimezone:
  # Earliest timestamp - Earliest allowable timestamp
  # [string]
  timestampEarliest:
```

```
# Latest timestamp - Latest allowable timestamp
# [string]
timestampLatest:
# Timestamp configuration - Timestamp extraction configuration
# [required]
timestamp:
  # Timestamp type - Type of timestamp extraction
  # One of: auto | format | current
  # [string; required]
  type:
  # Timestamp length - Length of timestamp to extract
  # [number]
  length:
  # Timestamp format - Format string for timestamp parsing
  # [string]
  format:
# Fields - Field definitions for extraction
fields:
# Event byte limit - The maximum number of bytes that an event can be before being
# [number; required]
maxEventBytes:
# Disabled - Whether this rule is disabled
# [boolean]
disabled:
# Parser enabled - Whether to enable parser for this rule
# [boolean]
parserEnabled:
# Parser configuration - Parser settings for this rule
parser:
# Should use data raw - Use the field in the data called _raw for post processors
# [boolean]
shouldUseDataRaw:
# Event Breaker - The regex used to break the stream into events at the beginning o
# [string]
eventBreakerRegex:
# JSON Array Field - Name of the Array field to parse events from (json_array type
# [string]
jsonArrayField:
# JSON Extract All - Whether json_array breaker should extract fields (json_array t
# [boolean]
jsonExtractAll:
# JSON Time Field - Name of the field to use as _time (json_array type only)
# [string]
```

```
jsonTimeField:
```

17.5.2. Certificates.Yml

`certificates.yml` maintains a list of configured certificates and their parameters.

certificates.yml

```
# Name - Unique identifier for this certificate
# [string; required]
id:
# Description - Optional description for the certificate
# [string]
description:
# Certificate - Drag/drop or upload host certificate in PEM/Base64 format, or paste i
# here
# [string; required]
cert:
# Private key - Certificate private key in PEM format
# [string; required]
privKey:
# Passphrase - Optional passphrase for the private key
# [string]
passphrase:
# CA certificate - Optionally, drag/drop or upload all CA certificates in PEM/Base64
# certificate contents here. Certificates can be used for client and/or server auther
# [string]
ca:
# Referenced - List of configurations that reference this certificate
inUse:
```

Copy

17.5.3. Cribl.Yml

`cribl.yml` contains settings for configuring API and other system properties.

cribl.yml

```
# API Server Settings
api:
  # Host - Hostname or address to bind API server to. Defaults to 0.0.0.0. Using $CRI
  # overrides this setting.
  # [string; default: 0.0.0.0; required]
  host:
  # Port - API port to listen to. Defaults to 9000. Using $CRIBL_API_PORT overrides t
  # [number; max: 65535; default: 9000; required]
  port:
  # Protocol - Protocol that API server speaks, defaults to http1.1
  # [string; default: http1.1]
  protocol:
  # Retry count - Number of times to retry binding to API port
  # [number; min: 0; default: 120]
  retryCount:
  # Retry period - Period, in seconds, between consecutive port binding retries
  # [number; min: 0; default: 5]
  retrySleepSecs:
  # URL base path - URL base path from which to serve all assets (useful when behind
  # [string; default: '']
  baseUrl:
  # Local UI access - Enable to allow direct browser access to the Cribl Edge Edge No
  # [boolean; default: false]
  disabled:
  # Listen on port - Expose the API service to the network on the configured port.
  # [boolean; default: true]
  listenOnPort:
  # Enable teleporting to Edge Nodes - Enables authenticated access to each Edge Node
  # Leader
  # [boolean; default: false]
  workerRemoteAccess:
  # Log out on roles change - Log users out when their roles change
  # [boolean; default: true]
  revokeOnRoleChange:
  # Auth token TTL - How long (in seconds) authentication tokens remain valid. Defaul
  # minimum is 1 sec.
  # [number; min: 1; default: 3600]
  authTokenTTL:
  # Session idle time limit - How long (in seconds) Cribl Edge will observe no user i
  # before invalidating user's session tokens. Default is 1 hr.; minimum is 60 sec.
  # [number; min: 60; default: 3600]
  idleSessionTTL:
```

```
  # Login rate limit - Rate limit, expressed as maximum number of requests per interv
  # minute, hour, day). Examples: 3/second, 5/minute
  # [string]
  loginRateLimit:
  # SSO/SLO callback rate limit - Rate limit for SSO and SLO callback endpoints. Expr
  # maximum number of requests per second, minute, hour, or day. Examples: 3/second,
  # When limit is reached, the Cribl Edge sends 429 Too Many Requests HTTP responses.
  # [string]
  ssoRateLimit:
  # HTTP headers - Custom HTTP headers to be sent with every response
  headers:
  apiCache:
    # Enable API cache - Enable caching of browser's frequent API requests. (Disablir
    # the response time.)
    # [boolean; default: false]
    disabled:
  # SSL - Enable SSL on API. Defaults to disabled.
  ssl:
    # Disabled
    # [boolean; default: true; required]
    disabled:
    # Certificate - The name of the predefined certificate
    # [string]
    certificateName:
    # Private key path - Path on server in which to find the private key to use. PEM
    # reference $ENV_VARS.
    # [string; required]
    privKeyPath:
    # Passphrase - Passphrase to use to decrypt private key
    # [string]
    passphrase:
    # Certificate path - Path on server in which to find certificates to use. PEM for
    # reference $ENV_VARS.
    # [string; required]
    certPath:
    # CA certificate path - Path on server where to find CA certificates to use. PEM
    # reference $ENV_VARS.
    # [string]
    caPath:
# Support
support:
  # Feature Flag Overrides
  featureFlagOverrides:
```

```
    # Flag ID
    # [string; min: 1; max: 100; required]
    flagId:
    # Enabled
    # [boolean; default: true; required]
    disabled:
# Authentication Settings
auth:
  # Type - Select from this list of supported authentication providers
  # One of: local | splunk | ldap | openid | saas | saml
  # [string; required]
  type:
  # LDAP servers - List of LDAP servers. Each entry should contain host:port (example
  # localhost:389).
  # [array; min: 1; required]
  ldapServers:
  # Bind DN - Distinguished Name of entity to authenticate with LDAP server. Example:
  # 'cn=admin,dc=example,dc=org'
  # [string; required]
  bindDN:
  # Password - Distinguished Name password used to authenticate with LDAP server
  # [string; required]
  bindCredentials:
  # Username field - LDAP user search field, such as cn or uid
  # [string; default: cn; required]
  usernameField:
  # User search base - Starting point to search LDAP for users. Example: 'dc=example,
  # [string; required]
  searchBase:
  # Secure - Enable to use a secure LDAP connection (ldaps://); disable for unsecure
  # connection
  # [boolean; default: false]
  secure:
  # User search filter - LDAP search filter to apply when finding user. Example:
  # (&(group=admin)(!(department=123*)))
  # [string]
  searchFilter:
  # Group search base - Starting point to search LDAP for groups. Example: 'dc=exampl
  # [string]
  groupSearchBase:
  # Group member field - LDAP group search field, such as member
  # [string; default: member]
  groupMemberField:
```

```
# Group membership attribute - Attribute name of LDAP user object, such as dn. Dete
# member attribute's value, which defines group's allowed users.
# [string; default: dn]
groupMemberValField:
# Group search filter - LDAP search filter to apply when finding group. Example:
# (&(cn=cribl*)(objectclass=group))
# [string]
groupSearchFilter:
# Group name field - LDAP group field, such as cn
# [string; default: cn]
groupField:
# Connection timeout (ms)
# [number; default: 5000]
connectTimeout:
# Validate server certs - Validate server certificates globally. Disable to allow s
# certificates.
# [boolean; default: true]
rejectUnauthorized:
# Host - Hostname or IP address of Splunk instance
# [string]
host:
# Port - Management port of Splunk instance
# [number; default: 8089]
port:
# Issuer - The OIDC provider's issuer URL, e.g., https://accounts.google.com
# [string]
issuer:
# Authorization URL - The OIDC provider's authorization URL
# [string]
authorizationURL:
# Token URL - The OIDC provider's token URL
# [string]
tokenURL:
# UserInfo URL - The OIDC provider's UserInfo URL
# [string]
userInfoURL:
# Client ID - The client ID provided by the OIDC provider
# [string]
clientID:
# Client Secret - The client secret provided by the OIDC provider
# [string]
clientSecret:
# Callback URL - The callback URL where the OIDC provider redirects after authentic
```

```
  # [string]
  callbackURL:
  # Scope - The scopes to request from the OIDC provider
  # [string]
  scope:
system:
  # One of: api | auto | false
  # [string]
  upgrade:
  # One of: api | false
  # [string]
  restart:
  # One of: splunk-app | standalone
  # [string]
  installType:
  # Share telemetry with Cribl - Help improve Cribl's products and capabilities by sh
  # anonymized telemetry data with Cribl
  # [boolean]
  intercom:
  rollback:
    # Enable automatic rollback - Automatically roll back the upgrade if it fails to
    # the Worker Node fails to connect to the Leader
    # [boolean; default: true; required]
    rollbackEnabled:
    # Rollback timeout (ms) - Time to wait before checking node health to perform an
    # rollback
    # [number; min: 1; default: 30000]
    rollbackTimeout:
    # Rollback condition retries - Number of times to retry health check before perfo
    # rollback
    # [number; min: 0; default: 5]
    rollbackRetries:
    # Check interval (ms) - Time to wait between retries
    # [number; min: 100; default: 1000]
    checkInterval:
  backups:
    # Backups directory - Directory where backups will be stored
    # [string; default: $CRIBL_STATE_DIR/backups]
    backupsDirectory:
    # Backup persistence - Relative time expression for how long to keep the backups.
    # [string; default: 24h]
    backupPersistence:
workers:
```

```
  # number of worker processes to spawn, if less than 1 the value is added to CPU cou
  # [number; default: 1]
  count:
  # [number; min: 1024; default: 2048]
  memory:
# Default TLS Settings
tls:
  # Minimum TLS version - Minimum TLS version. Defaults to TLS 1.2.
  # One of: TLSv1 | TLSv1.1 | TLSv1.2 | TLSv1.3
  # [string]
  minVersion:
  # Maximum TLS version - Maximum TLS version. Defaults to TLS 1.3.
  # One of: TLSv1 | TLSv1.1 | TLSv1.2 | TLSv1.3
  # [string]
  maxVersion:
  # Default cipher list - Default suite of enabled and disabled TLS ciphers. Defaults
  # ECDHE-RSA-AES128-GCM-SHA256:
  # ECDHE-ECDSA-AES128-GCM-SHA256:
  # ECDHE-RSA-AES256-GCM-SHA384:
  # ECDHE-ECDSA-AES256-GCM-SHA384:
  # DHE-RSA-AES128-GCM-SHA256:
  # ECDHE-RSA-AES128-SHA256:
  # DHE-RSA-AES128-SHA256:
  # ECDHE-RSA-AES256-SHA384:
  # DHE-RSA-AES256-SHA384:
  # ECDHE-RSA-AES256-SHA256:
  # DHE-RSA-AES256-SHA256:
  # HIGH:
  # !aNULL:
  # !eNULL:
  # !EXPORT:
  # !DES:
  # !RC4:
  # !MD5:
  # !PSK:
  # !SRP:
  # !CAMELLIA
  # [string]
  defaultCipherList:
  # ECDH curve - The curve name, or a colon-separated list of curve NIDs or names, to
  # key agreement. For example: 'P-521:P-384:P-256'. Defaults to 'auto'.
  # [string; default: auto]
  defaultEcdhCurve:
```

```
  # Validate server certs - Validate server certificates globally. Disable to allow s
  # certificates.
  # [boolean; default: true]
  rejectUnauthorized:
# Proxy Settings
proxy:
  # Use proxy env vars - Honor HTTP_PROXY/HTTPS_PROXY environment variables
  # [boolean; default: true; required]
  useEnvVars:
git:
  # Branch - The branch to track in your Stream deployment's git repository
  # [string; default: master]
  branch:
  # GitOps workflow - The GitOps workflow for managing Cribl configuration
  # One of: none | push
  # [string; default: none]
  gitOps:
  # Collapse actions - Collapse Commit & Deploy into a single action. If you've confi
  # remote, Commit & Git Push will also be collapsed. Your default commit message bel
  # used for all commits.
  # [boolean]
  commitDeploySingleAction:
  # Default commit message - Enter a default message to use for all commits
  # [string; required]
  defaultCommitMessage:
  # Remote URL
  # [string]
  remote:
  # Git authentication type
  # One of: none | ssh | basic
  # [string; default: ssh]
  authType:
  # Scheduled global actions - Global git actions to run automatically on a schedule
  # One of: none | commit | push | commitPush
  # [string; default: none]
  autoAction:
  # Git timeout - Maximum time (in milliseconds) to wait for git processes before enc
  # Enter 0 to wait indefinitely.
  # [number; min: 0; default: 600000]
  timeout:
  # Generate commit messages automatically with Cribl Copilot - Cribl Copilot-generat
  # override any default commit message you may have set
  # [boolean]
```

```
  copilotAutoGitCommitMessages:
  # SSH private key - SSH private key (without passphrase) to use for authentication
  # repo
  # [string]
  sshKey:
  # SSH strict host key checking - Validate key against known hosts, to prevent spoof
  # impersonation attacks. For details, see "Verifying Host Keys" [on the Linux man
  # page](https://linux.die.net/man/1/ssh).
  # [boolean; default: true]
  strictHostKeyChecking:
  # Username - Username for authentication on remote git repo
  # [string; required]
  user:
  # Password/Token - Password or Personal Access Token for authentication on remote g
  # [string; required]
  password:
# [boolean]
fips:
# SNI Routing
sni:
  # Disable SNI-based connection routing - Advanced setting - do not change without c
  # Cribl Support. This setting affects how connections are routed within the Cribl c
  # Changing this setting could affect the scalability of your system.
  # [boolean; default: false]
  disableSNIRouting:
# Sockets for IPC
sockets:
  # Directory - Directory to hold sockets for inter-process communication (IPC), such
  # balancing and Worker Processes. Defaults to your system's temp directory.
  # [string; max: 64]
  directory:
# Shutdown Settings
shutdown:
  # Drain timeout (sec) - How many seconds to wait for writes to complete before Crib
  # down on Workers
  # [number; min: 1; max: 600; default: 10]
  drainTimeout:
# PII Detection
pii:
  # Enable PII detection - Enable periodic detection of PII in this group
  # [boolean; default: false]
  enablePiiDetection:
# Upgrade Group Settings
```

```
  upgradeGroupSettings:
    # Quantity - Percentage of workers to upgrade
    # [number; default: 100]
    quantity:
    # Rolling upgrade - Upgrade workers one at a time
    # [boolean; default: true]
    isRolling:
    # Retry delay (ms) - Number of milliseconds to wait in between upgrade attempts
    # [number; default: 1000]
    retryDelay:
    # Retry count - Number of times to retry failed upgrade
    # [number; default: 5]
    retryCount:
  # Upgrade Settings
  upgradeSettings:
    # Package source - Select download package source
    # One of: cdn | path
    # [string; default: cdn]
    upgradeSource:
    # Enable automatic upgrades - Disable to opt out of automatic upgrades when Cribl r
    # versions
    # [boolean; default: true]
    disableAutomaticUpgrade:
    # Enable Legacy Edge upgrades - Opt in legacy upgrades
    # [boolean; default: false]
    enableLegacyEdgeUpgrade:
    # Upgrade check period - Interval over which to check for out-of-date Nodes
    # [string; default: 1h; required]
    automaticUpgradeCheckPeriod:
    # Custom Path - Provide your own URLs or local paths for platform-specific Cribl pa
    # [required]
    packageUrls:
      # Platform-Specific Package Location - Package HTTP URL or local path
      # [string; min: 1; required]
      packageUrl:
      # Package Hash Location - Package's MD5 or SHA256 hash HTTP URL or local path
      # [string; min: 1]
      packageHashUrl:
```

Copy

Example `cribl.yml`:

$CRIBL_HOME/default/cribl/cribl.yml

```
api:
  host: 0.0.0.0
  port: 9000
```

17.5.4. Groups.Yml

`groups.yml` maintains a list of groups and their configuration versions.

groups.yml

```
# Worker Group type - Whether this group accepts Hybrid (on-prem/customer-managed) nc
# Cribl-managed Cribl.Cloud nodes. (This field is only available in Cribl.Cloud.)
# [boolean; default: true]
onPrem:
# ID - Unique identifier for the group
# [string; required]
id:
# Enable teleporting to nodes - Enable remote access to nodes in this group
# [boolean; default: false]
workerRemoteAccess:
```

17.5.5. Inputs.Yml

`inputs.yml` contains settings for configuring inputs into Cribl.

inputs.yml

```
inputs: # [object]
  collection_input: # [object]
    type: # [string] Input Type
    breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event-br
    staleChannelFlushMs: # [number; minimum: 10, maximum: 43200000] Event Breaker buf
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Use QuickConnect - Direct connections to Destinations, and
      - pipeline: # [string] Pipeline or Pack
        output: # [string] Destination

    # ------------------------------------------------------

    preprocess: # [object]
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      command: # [string] Command - Command to feed the data through (via stdin) and
      args: # [array of strings] Arguments - Arguments to be added to the custom comm

      # ------------------------------------------------------

    throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to throttl
    metadata: # [array] Fields - Fields to add to events from this input
      - name: # [string] Field Name
        value: # [string] Value - JavaScript expression to compute field's value, enc
    disabled: # [boolean] Disabled
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
    environment: # [string] Environment - Optionally, enable this config only on a sp
    pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
      maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
      commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
      maxFileSize: # [string] File size limit - The maximum size to store in each que
      maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
```

```
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # ------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
kafka_input: # [object]
  type: # [string] Input Type
  brokers: # [array of strings] Bootstrap servers - Enter each Kafka bootstrap serv
  topics: # [array of strings] [required] Topic - Topic to subscribe to. Warning: T
  groupId: # [string] Group ID - The consumer group to which this instance belongs.
  fromBeginning: # [boolean] From beginning - Leave enabled if you want the Source,
  kafkaSchemaRegistry: # [object] Kafka Schema Registry Authentication
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    schemaRegistryURL: # [string] Schema Registry URL - URL for accessing the Confl
    schemaType: # [string] Schema type - The schema format used to encode and decod
    connectionTimeout: # [number; minimum: 1000, maximum: 60000] Connection timeout
    requestTimeout: # [number; minimum: 1000, maximum: 60000] Request timeout (ms)
    maxRetries: # [number; maximum: 100] Retry limit - Maximum number of times to t
    auth: # [object]  - Credentials to use when authenticating with the schema regi
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      credentialsSecret: # [string] Credentials secret - Select or create a secret

      # ------------------------------------------------------

    tls: # [object] TLS settings (client side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      rejectUnauthorized: # [boolean] Validate server certs - Reject certificates t
                  trusted CA (such as the system's). Defaults to Enabled. Overrides
      servername: # [string] Server name (SNI) - Server name for the SNI (Server Na
      certificateName: # [string] Certificate - The name of the predefined certific
      caPath: # [string] CA certificate path - Path on client in which to find CA c
      privKeyPath: # [string] Private key path (mutual auth) - Path on client in wh
      certPath: # [string] Certificate path (mutual auth) - Path on client in which
```

```
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
    minVersion: # [string] Minimum TLS version
    maxVersion: # [string] Maximum TLS version

    # ------------------------------------------------------


  # --------------------------------------------------------

connectionTimeout: # [number; minimum: 1000, maximum: 3600000] Connection timeout
requestTimeout: # [number; minimum: 1000, maximum: 3600000] Request timeout (ms)
maxRetries: # [number; maximum: 100] Retry limit - If messages are failing, you c
maxBackOff: # [number; minimum: 30000, maximum: 180000] Backoff limit (ms) - The
initialBackoff: # [number; minimum: 300, maximum: 600000] Initial retry interval
backoffRate: # [number; minimum: 2, maximum: 20] Backoff multiplier - Set the bac
authenticationTimeout: # [number; minimum: 1000, maximum: 3600000] Authentication
reauthenticationThreshold: # [number; minimum: 1000, maximum: 1800000] Reauthenti
sasl: # [object] Authentication - Authentication parameters to use when connectir
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  mechanism: # [string] SASL mechanism
  oauthEnabled: # [boolean] Enable OAuth - Enable OAuth authentication

  # ------------------------------------------------------

tls: # [object] TLS settings (client side)
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  rejectUnauthorized: # [boolean] Validate server certs - Reject certificates tha
                trusted CA (such as the system's). Defaults to Enabled. Overrides
  servername: # [string] Server name (SNI) - Server name for the SNI (Server Name
  certificateName: # [string] Certificate - The name of the predefined certificat
  caPath: # [string] CA certificate path - Path on client in which to find CA cer
  privKeyPath: # [string] Private key path (mutual auth) - Path on client in whic
  certPath: # [string] Certificate path (mutual auth) - Path on client in which t
  passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
  minVersion: # [string] Minimum TLS version
  maxVersion: # [string] Maximum TLS version
```

```
  # ------------------------------------------------------

sessionTimeout: # [number; minimum: 1000, maximum: 3600000] Session timeout (ms)
  Timeout used to detect client failures when using Kafka's group-management faci
  If the client sends no heartbeats to the broker before the timeout expires,
  the broker will remove the client from the group and initiate a rebalance.
  Value must be between the broker's configured group.min.session.timeout.ms and
  See [Kafka's documentation](https://kafka.apache.org/documentation/#consumercor
rebalanceTimeout: # [number; minimum: 1000, maximum: 3600000] Rebalance timeout (
  Maximum allowed time for each worker to join the group after a rebalance begins
  If the timeout is exceeded, the coordinator broker will remove the worker from
  See [Kafka's documentation](https://kafka.apache.org/documentation/#connectconf
heartbeatInterval: # [number; minimum: 1000, maximum: 3600000] Heartbeat interval
  Expected time between heartbeats to the consumer coordinator when using Kafka's
  Value must be lower than sessionTimeout and typically should not exceed 1/3 of
  See [Kafka's documentation](https://kafka.apache.org/documentation/#consumercor
autoCommitInterval: # [number; minimum: 1000, maximum: 3600000] Offset commit int
autoCommitThreshold: # [number; minimum: 1, maximum: 10000] Offset commit thresho
maxBytesPerPartition: # [number; minimum: 1, maximum: 10000000] Byte limit, per p
maxBytes: # [number; minimum: 1, maximum: 1000000000] Byte limit - Maximum number
maxSocketErrors: # [number; maximum: 100] Error limit, per socket - Maximum numbe
metadata: # [array] Fields - Fields to add to events from this input
  - name: # [string] Field Name
    value: # [string] Value - JavaScript expression to compute field's value, enc
description: # [string] Description
disabled: # [boolean] Disabled
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

# -------------- if sendToRoutes is false ---------------

connections: # [array] Use QuickConnect - Direct connections to Destinations, and
  - pipeline: # [string] Pipeline or Pack
    output: # [string] Destination

# ------------------------------------------------------

environment: # [string] Environment - Optionally, enable this config only on a sp
pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

# -------------- if pqEnabled is true ---------------

pq: # [object]
```

```
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # ------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
msk_input: # [object]
  type: # [string] Input Type
  brokers: # [array of strings] Bootstrap servers - Enter each Kafka bootstrap serv
  topics: # [array of strings] [required] Topic - Topic to subscribe to. Warning: T
  groupId: # [string] Group ID - The consumer group to which this instance belongs.
  fromBeginning: # [boolean] From beginning - Leave enabled if you want the Source,
  sessionTimeout: # [number; minimum: 1000, maximum: 3600000] Session timeout (ms)
    Timeout used to detect client failures when using Kafka's group-management faci
    If the client sends no heartbeats to the broker before the timeout expires,
    the broker will remove the client from the group and initiate a rebalance.
    Value must be between the broker's configured group.min.session.timeout.ms and
    See [Kafka's documentation](https://kafka.apache.org/documentation/#consumercor
  rebalanceTimeout: # [number; minimum: 1000, maximum: 3600000] Rebalance timeout (
    Maximum allowed time for each worker to join the group after a rebalance begins
    If the timeout is exceeded, the coordinator broker will remove the worker from
    See [Kafka's documentation](https://kafka.apache.org/documentation/#connectconf
  heartbeatInterval: # [number; minimum: 1000, maximum: 3600000] Heartbeat interval
    Expected time between heartbeats to the consumer coordinator when using Kafka's
    Value must be lower than sessionTimeout and typically should not exceed 1/3 of
    See [Kafka's documentation](https://kafka.apache.org/documentation/#consumercor
  metadata: # [array] Fields - Fields to add to events from this input
    - name: # [string] Field Name
      value: # [string] Value - JavaScript expression to compute field's value, enc
  kafkaSchemaRegistry: # [object] Kafka Schema Registry Authentication
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    schemaRegistryURL: # [string] Schema Registry URL - URL for accessing the Confl
    schemaType: # [string] Schema type - The schema format used to encode and decod
    connectionTimeout: # [number; minimum: 1000, maximum: 60000] Connection timeout
    requestTimeout: # [number; minimum: 1000, maximum: 60000] Request timeout (ms)
```

```
  maxRetries: # [number; maximum: 100] Retry limit - Maximum number of times to t
  auth: # [object]  - Credentials to use when authenticating with the schema regi
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    credentialsSecret: # [string] Credentials secret - Select or create a secret

    # ----------------------------------------------------

  tls: # [object] TLS settings (client side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    rejectUnauthorized: # [boolean] Validate server certs - Reject certificates t
                trusted CA (such as the system's). Defaults to Enabled. Overrides
    servername: # [string] Server name (SNI) - Server name for the SNI (Server Na
    certificateName: # [string] Certificate - The name of the predefined certific
    caPath: # [string] CA certificate path - Path on client in which to find CA c
    privKeyPath: # [string] Private key path (mutual auth) - Path on client in wh
    certPath: # [string] Certificate path (mutual auth) - Path on client in which
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
    minVersion: # [string] Minimum TLS version
    maxVersion: # [string] Maximum TLS version

    # ----------------------------------------------------


  # ------------------------------------------------------

connectionTimeout: # [number; minimum: 1000, maximum: 3600000] Connection timeout
requestTimeout: # [number; minimum: 1000, maximum: 3600000] Request timeout (ms)
maxRetries: # [number; maximum: 100] Retry limit - If messages are failing, you c
maxBackOff: # [number; minimum: 30000, maximum: 180000] Backoff limit (ms) - The
initialBackoff: # [number; minimum: 300, maximum: 600000] Initial retry interval
backoffRate: # [number; minimum: 2, maximum: 20] Backoff multiplier - Set the bac
authenticationTimeout: # [number; minimum: 1000, maximum: 3600000] Authentication
reauthenticationThreshold: # [number; minimum: 1000, maximum: 1800000] Reauthenti
awsAuthenticationMethod: # [string] [required] Authentication method - AWS auther

# -------------- if awsAuthenticationMethod is manual ---------------
```

```
awsApiKey: # [string] Access key

# ------------------------------------------------------


# -------------- if awsAuthenticationMethod is secret ---------------

awsSecret: # [string] Secret key pair - Select or create a stored secret that ref

# ------------------------------------------------------

awsSecretKey: # [string] Secret key
region: # [string] [required] Region - Region where the MSK cluster is located
endpoint: # [string] Endpoint - MSK cluster service endpoint. If empty, defaults
signatureVersion: # [string] Signature version - Signature version to use for sig
reuseConnections: # [boolean] Reuse connections - Reuse connections between reque
rejectUnauthorized: # [boolean] Reject unauthorized certificates - Reject certifi
enableAssumeRole: # [boolean] Enable for MSK - Use Assume Role credentials to acc
assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the role
assumeRoleExternalId: # [string] External ID - External ID to use when assuming r
durationSeconds: # [number; minimum: 900, maximum: 43200] Duration (seconds) - Du
tls: # [object] TLS settings (client side)
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  rejectUnauthorized: # [boolean] Validate server certs - Reject certificates tha
                trusted CA (such as the system's). Defaults to Enabled. Overrides
  servername: # [string] Server name (SNI) - Server name for the SNI (Server Name
  certificateName: # [string] Certificate - The name of the predefined certificat
  caPath: # [string] CA certificate path - Path on client in which to find CA cer
  privKeyPath: # [string] Private key path (mutual auth) - Path on client in whic
  certPath: # [string] Certificate path (mutual auth) - Path on client in which t
  passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
  minVersion: # [string] Minimum TLS version
  maxVersion: # [string] Maximum TLS version

  # ------------------------------------------------------

autoCommitInterval: # [number; minimum: 1000, maximum: 3600000] Offset commit int
autoCommitThreshold: # [number; minimum: 1, maximum: 10000] Offset commit thresho
maxBytesPerPartition: # [number; minimum: 1, maximum: 10000000] Byte limit, per p
maxBytes: # [number; minimum: 1, maximum: 1000000000] Byte limit - Maximum numbe
```

```
  maxSocketErrors: # [number; maximum: 100] Error limit, per socket - Maximum numbe
  description: # [string] Description
  disabled: # [boolean] Disabled
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Use QuickConnect - Direct connections to Destinations, and
    - pipeline: # [string] Pipeline or Pack
      output: # [string] Destination

  # --------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a sp
  pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # --------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
http_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled
  host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses
  port: # [number; maximum: 65535] [required] Port - Port to listen on
  authTokens: # [array of strings] Auth tokens - Shared secrets to be provided by a
  tls: # [object] TLS settings (server side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    certificateName: # [string] Certificate - The name of the predefined certificat
```

```
  privKeyPath: # [string] Private key path - Path on server containing the privat
  passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
  certPath: # [string] Certificate path - Path on server containing certificates
  caPath: # [string] CA certificate path - Path on server containing CA certifica
  requestCert: # [boolean] Authenticate client (mutual auth) - Require clients to
  minVersion: # [string] Minimum TLS version
  maxVersion: # [string] Maximum TLS version

  # ------------------------------------------------------

maxActiveReq: # [number] Active request limit - Maximum number of active requests
maxRequestsPerSocket: # [integer] Requests-per-socket limit - Maximum number of r
enableProxyHeader: # [boolean] Show originating IP - Extract the client IP and po
captureHeaders: # [boolean] Capture request headers - Add request headers to ever
activityLogSampleRate: # [number; minimum: 1] Activity log sample rate - How ofte
requestTimeout: # [number] Request timeout (seconds) - How long to wait for an in
socketTimeout: # [number] Socket timeout (seconds) - How long Cribl Edge should w
keepAliveTimeout: # [number; minimum: 1, maximum: 600] Keep-alive timeout (second
enableHealthCheck: # [boolean] Health check endpoint - Expose the /cribl_health e
ipAllowlistRegex: # [string] IP allowlist regex - Messages from matched IP addres
ipDenylistRegex: # [string] IP denylist regex - Messages from matched IP addresse
criblAPI: # [string] Cribl HTTP event API - Absolute path on which to listen for
elasticAPI: # [string] Elasticsearch API endpoint (Bulk API) - Absolute path on w
splunkHecAPI: # [string] Splunk HEC endpoint - Absolute path on which listen for
splunkHecAcks: # [boolean] Enable Splunk HEC acknowledgements
metadata: # [array] Fields - Fields to add to events from this input
  - name: # [string] Field Name
    value: # [string] Value - JavaScript expression to compute field's value, enc
authTokensExt: # [array] Auth tokens - Shared secrets to be provided by any clien
  - token: # [string] Token - Shared secret to be provided by any client (Authori
    description: # [string] Description
    metadata: # [array] Fields - Fields to add to events referencing this token
    - name: # [string] Field Name
      value: # [string] Value - JavaScript expression to compute field's value, e
description: # [string] Description
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

# -------------- if sendToRoutes is false ---------------

connections: # [array] Use QuickConnect - Direct connections to Destinations, and
  - pipeline: # [string] Pipeline or Pack
    output: # [string] Destination
```

```
  # ------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a sp
  pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # ------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
splunk_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled
  host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses
  port: # [number; maximum: 65535] [required] Port - Port to listen on
  tls: # [object] TLS settings (server side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    certificateName: # [string] Certificate - The name of the predefined certificat
    privKeyPath: # [string] Private key path - Path on server containing the privat
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
    certPath: # [string] Certificate path - Path on server containing certificates
    caPath: # [string] CA certificate path - Path on server containing CA certifica
    requestCert: # [boolean] Authenticate client (mutual auth) - Require clients to
    minVersion: # [string] Minimum TLS version
    maxVersion: # [string] Maximum TLS version

    # ------------------------------------------------------

  ipWhitelistRegex: # [string] IP allowlist regex - Regex matching IP addresses tha
  maxActiveCxn: # [number] Active connection limit - Maximum number of active conne
```

```
socketIdleTimeout: # [number] Socket idle timeout (seconds) - How long Cribl Edge
socketEndingMaxWait: # [number] Forced socket termination timeout (seconds) - How
socketMaxLifespan: # [number] Socket max lifespan (seconds) - The maximum duratic
enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection i
metadata: # [array] Fields - Fields to add to events from this input
  - name: # [string] Field Name
    value: # [string] Value - JavaScript expression to compute field's value, enc
breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event-br
staleChannelFlushMs: # [number; minimum: 10, maximum: 43200000] Event Breaker buf
authTokens: # [array] Auth tokens - Shared secrets to be provided by any Splunk f
  - token: # [string] Token - Shared secrets to be provided by any Splunk forwarc
    description: # [string] Description
maxS2Sversion: # [string] Max S2S version - The highest S2S protocol version to a

# -------------- if maxS2Sversion is v4 ---------------

useFwdTimezone: # [boolean] Use Universal Forwarder time zone - Event Breakers wi
dropControlFields: # [boolean] Drop control fields - Drop Splunk control fields s
extractMetrics: # [boolean] Extract metrics - Extract and process Splunk-generate
compress: # [string] Compression - Controls whether to support reading compressed

# ------------------------------------------------------

description: # [string] Description
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

# -------------- if sendToRoutes is false ---------------

connections: # [array] Use QuickConnect - Direct connections to Destinations, and
  - pipeline: # [string] Pipeline or Pack
    output: # [string] Destination

# ------------------------------------------------------

environment: # [string] Environment - Optionally, enable this config only on a sp
pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

# -------------- if pqEnabled is true ---------------

pq: # [object]
  mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
  maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
```

```
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # ------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
splunk_search_input: # [object]
  type: # [string] Input Type
  searchHead: # [string] Search head - Search head base URL. Can be an expression.
  search: # [string] [required] Search - Enter Splunk search here. Examples: 'index
  earliest: # [string] Earliest - The earliest time boundary for the search. Can be
  latest: # [string] Latest - The latest time boundary for the search. Can be an ex
  cronSchedule: # [string] [required] Cron schedule - A cron schedule on which to r
  endpoint: # [string] [required] Search endpoint - REST API used to create a searc
  outputMode: # [string] [required] Output mode - Format of the returned output
  endpointParams: # [array] Endpoint parameters - Optional request parameters to se
    - name: # [string] Parameter Name
      value: # [string] Value - JavaScript expression to compute the parameter's va
  endpointHeaders: # [array] Endpoint headers - Optional request headers to send to
    - name: # [string] Header Name
      value: # [string] Value - JavaScript expression to compute the header's value
  logLevel: # [string] Log level - Collector runtime log level (verbosity)
  requestTimeout: # [number; maximum: 2400] Request timeout (seconds) - HTTP reques
  useRoundRobinDns: # [boolean] Round-robin DNS - When a DNS server returns multipl
  rejectUnauthorized: # [boolean] Reject unauthorized certificates - Reject certifi
  encoding: # [string] Encoding - Character encoding to use when parsing ingested d
  keepAliveTime: # [number; minimum: 10] Keep alive time (seconds) - How often work
  jobTimeout: # [string] Job timeout - Maximum time the job is allowed to run (e.g.
  maxMissedKeepAlives: # [number; minimum: 2] Worker timeout (periods) - The number
  ttl: # [string] Time to live - Time to keep the job's artifacts on disk after job
  ignoreGroupJobsLimit: # [boolean] Ignore Worker Group job limits - When enabled,
  metadata: # [array] Fields - Fields to add to events from this input
    - name: # [string] Field Name
      value: # [string] Value - JavaScript expression to compute field's value, enc
  retryRules: # [object]
    type: # [string] Retry type - The algorithm to use when performing HTTP retries

    # -------------- if type is static ---------------

    codes: # [array of numbers] Retry HTTP codes - List of HTTP codes that trigger
```

```
  interval: # [number; maximum: 20000] Wait (ms) - Time interval between failed r
  limit: # [number; maximum: 20] Retry limit - The maximum number of times to ret
  enableHeader: # [boolean] Honor Retry-After header - Honor any Retry-After head
  retryConnectTimeout: # [boolean] Retry connection timeout - Make a single retry
  retryConnectReset: # [boolean] Retry connection reset - Retry request when a co

  # ------------------------------------------------------


  # -------------- if type is backoff ---------------

  interval: # [number; maximum: 20000] Initial retry interval (ms) - Time interva
  limit: # [number; maximum: 20] Retry limit - The maximum number of times to ret
  multiplier: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base for e
  codes: # [array of numbers] Retry HTTP codes - List of HTTP codes that trigger
  enableHeader: # [boolean] Honor Retry-After header - Honor any Retry-After head
  retryConnectTimeout: # [boolean] Retry connection timeout - Make a single retry
  retryConnectReset: # [boolean] Retry connection reset - Retry request when a co

  # ------------------------------------------------------

breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event-br
staleChannelFlushMs: # [number; minimum: 10, maximum: 43200000] Event Breaker buf
authType: # [string] Authentication type - Splunk Search authentication type

# -------------- if authType is basic ---------------

username: # [string] Username
password: # [string] Password

# ------------------------------------------------------


# -------------- if authType is token ---------------

token: # [string] Token - Bearer token to include in the authorization header

# ------------------------------------------------------


# -------------- if authType is credentialsSecret ---------------

credentialsSecret: # [string] Credentials secret - Select or create a secret that
```

```
# ------------------------------------------------------


# -------------- if authType is textSecret ---------------

textSecret: # [string] Token (text secret) - Select or create a stored text secre

# ------------------------------------------------------


# -------------- if authType is oauth ---------------

loginUrl: # [string] Login URL - URL for OAuth
secretParamName: # [string] OAuth Secret parameter name - Secret parameter name t
secret: # [string] OAuth secret - Secret parameter value to pass in request body
tokenAttributeName: # [string] Token attribute name - Name of the auth token attr
spacer: # [null]
authHeaderExpr: # [string] Authorize expression - JavaScript expression to comput
tokenTimeoutSecs: # [number; minimum: 1, maximum: 300000] Refresh interval (secs.
oauthParams: # [array] OAuth parameters - Additional parameters to send in the OA
  - name: # [string] Name - OAuth parameter name
    value: # [string] Value - OAuth parameter value
oauthHeaders: # [array] OAuth headers - Additional headers to send in the OAuth t
  - name: # [string] Name - OAuth header name
    value: # [string] Value - OAuth header value

# ------------------------------------------------------

description: # [string] Description
disabled: # [boolean] Disabled
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

# -------------- if sendToRoutes is false ---------------

connections: # [array] Use QuickConnect - Direct connections to Destinations, and
  - pipeline: # [string] Pipeline or Pack
    output: # [string] Destination

# ------------------------------------------------------

environment: # [string] Environment - Optionally, enable this config only on a sp
```

```
  pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # ---------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
splunk_hec_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled
  host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses
  port: # [number; maximum: 65535] [required] Port - Port to listen on
  authTokens: # [array] Auth tokens - Shared secrets to be provided by any client (
    - authType: # [string] Authentication method - Select Manual to enter an auth t
      enabled: # [boolean] Enable token
      description: # [string] Description - Optional token description
      allowedIndexesAtToken: # [array of strings] Allowed indexes - Enter the value
      metadata: # [array] Fields - Fields to add to events referencing this token
      - name: # [string] Field Name
        value: # [string] Value - JavaScript expression to compute field's value, e
  tls: # [object] TLS settings (server side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    certificateName: # [string] Certificate - The name of the predefined certificat
    privKeyPath: # [string] Private key path - Path on server containing the privat
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
    certPath: # [string] Certificate path - Path on server containing certificates
    caPath: # [string] CA certificate path - Path on server containing CA certifica
    requestCert: # [boolean] Authenticate client (mutual auth) - Require clients to
    minVersion: # [string] Minimum TLS version
    maxVersion: # [string] Maximum TLS version
```

```
  # ------------------------------------------------------

maxActiveReq: # [number] Active request limit - Maximum number of active requests
maxRequestsPerSocket: # [integer] Requests-per-socket limit - Maximum number of r
enableProxyHeader: # [boolean] Show originating IP - Extract the client IP and po
captureHeaders: # [boolean] Capture request headers - Add request headers to ever
activityLogSampleRate: # [number; minimum: 1] Activity log sample rate - How ofte
requestTimeout: # [number] Request timeout (seconds) - How long to wait for an ir
socketTimeout: # [number] Socket timeout (seconds) - How long Cribl Edge should w
keepAliveTimeout: # [number; minimum: 1, maximum: 600] Keep-alive timeout (second
ipAllowlistRegex: # [string] IP allowlist regex - Messages from matched IP addres
ipDenylistRegex: # [string] IP denylist regex - Messages from matched IP addresse
splunkHecAPI: # [string] [required] Splunk HEC endpoint - Absolute path on which
metadata: # [array] Fields - Fields to add to every event. Overrides fields added
  - name: # [string] Field Name
    value: # [string] Value - JavaScript expression to compute field's value, enc
allowedIndexes: # [array of strings] Allowed indexes - List values allowed in HEC
splunkHecAcks: # [boolean] Splunk HEC acks - Enable Splunk HEC acknowledgements
breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event-br
staleChannelFlushMs: # [number; minimum: 10, maximum: 43200000] Event Breaker buf
useFwdTimezone: # [boolean] Use Universal Forwarder time zone (S2S only) - Event
dropControlFields: # [boolean] Drop control fields (S2S only) - Drop Splunk contr
extractMetrics: # [boolean] Extract metrics (S2S only) - Extract and process Splu
accessControlAllowOrigin: # [array of strings] CORS allowed origins - Optionally,
accessControlAllowHeaders: # [array of strings] CORS allowed headers - Optionally
emitTokenMetrics: # [boolean] Emit per-token request metrics - Emit per-token (<p
description: # [string] Description
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

# -------------- if sendToRoutes is false ---------------

connections: # [array] Use QuickConnect - Direct connections to Destinations, and
  - pipeline: # [string] Pipeline or Pack
    output: # [string] Destination

# -------------------------------------------------------

environment: # [string] Environment - Optionally, enable this config only on a sp
pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

# -------------- if pqEnabled is true ---------------
```

```
  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # ------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
azure_blob_input: # [object]
  type: # [string] Input Type
  queueName: # [string] Queue - The storage account queue name blob notifications w
  fileFilter: # [string] Filename filter - Regex matching file names to download ar
  visibilityTimeout: # [number; maximum: 604800] Visibility timeout (secs) - The du
  numReceivers: # [number; minimum: 1, maximum: 100] Number of receivers - How many
  maxMessages: # [number; minimum: 1, maximum: 32] Message limit - The maximum numb
  servicePeriodSecs: # [number; minimum: 1, maximum: 10] Service period (secs) - Th
  skipOnError: # [boolean] Skip file on error - Skip files that trigger a processin
  metadata: # [array] Fields - Fields to add to events from this input
    - name: # [string] Field Name
      value: # [string] Value - JavaScript expression to compute field's value, enc
  breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event-br
  staleChannelFlushMs: # [number; minimum: 10, maximum: 43200000] Event Breaker buf
  parquetChunkSizeMB: # [number; minimum: 1, maximum: 100] Parquet chunk size limit
  parquetChunkDownloadTimeout: # [number; minimum: 1, maximum: 3600] Parquet chunk
  authType: # [string] Authentication method

  # -------------- if authType is manual ---------------

  connectionString: # [string] Connection string - Enter your Azure Storage account

  # ------------------------------------------------------


  # -------------- if authType is secret ---------------

  textSecret: # [string] Connection string (text secret) - Select or create a store

  # ------------------------------------------------------
```

```
# -------------- if authType is clientSecret ---------------

storageAccountName: # [string] Storage account name - The name of your Azure stor
tenantId: # [string] Tenant ID - The service principal's tenant ID
clientId: # [string] Client ID - The service principal's client ID
azureCloud: # [string] Azure Cloud - The Azure cloud to use. Defaults to Azure Pu
endpointSuffix: # [string] Endpoint suffix - Endpoint suffix for the service URL.
clientTextSecret: # [string] Client secret (text secret) - Select or create a stc

# ----------------------------------------------------------


# -------------- if authType is clientCert ---------------

storageAccountName: # [string] Storage account name - The name of your Azure stor
tenantId: # [string] Tenant ID - The service principal's tenant ID
clientId: # [string] Client ID - The service principal's client ID
azureCloud: # [string] Azure Cloud - The Azure cloud to use. Defaults to Azure Pu
endpointSuffix: # [string] Endpoint suffix - Endpoint suffix for the service URL.
certificate: # [object]
  certificateName: # [string] Certificate - The certificate you registered as cre

# ----------------------------------------------------------

description: # [string] Description
disabled: # [boolean] Disabled
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

# -------------- if sendToRoutes is false ---------------

connections: # [array] Use QuickConnect - Direct connections to Destinations, and
  - pipeline: # [string] Pipeline or Pack
    output: # [string] Destination

# ----------------------------------------------------------

environment: # [string] Environment - Optionally, enable this config only on a sp
pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

# -------------- if pqEnabled is true ---------------
```

```
  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # ------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
elastic_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled
  host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses
  port: # [number; maximum: 65535] [required] Port - Port to listen on
  tls: # [object] TLS settings (server side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    certificateName: # [string] Certificate - The name of the predefined certificat
    privKeyPath: # [string] Private key path - Path on server containing the privat
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
    certPath: # [string] Certificate path - Path on server containing certificates
    caPath: # [string] CA certificate path - Path on server containing CA certifica
    requestCert: # [boolean] Authenticate client (mutual auth) - Require clients to
    minVersion: # [string] Minimum TLS version
    maxVersion: # [string] Maximum TLS version

    # ------------------------------------------------------

  maxActiveReq: # [number] Active request limit - Maximum number of active requests
  maxRequestsPerSocket: # [integer] Requests-per-socket limit - Maximum number of r
  enableProxyHeader: # [boolean] Show originating IP - Extract the client IP and po
  captureHeaders: # [boolean] Capture request headers - Add request headers to even
  activityLogSampleRate: # [number; minimum: 1] Activity log sample rate - How ofte
  requestTimeout: # [number] Request timeout (seconds) - How long to wait for an in
  socketTimeout: # [number] Socket timeout (seconds) - How long Cribl Edge should w
  keepAliveTimeout: # [number; minimum: 1, maximum: 600] Keep-alive timeout (second
  enableHealthCheck: # [boolean] Health check endpoint - Expose the /cribl_health e
  ipAllowlistRegex: # [string] IP allowlist regex - Messages from matched IP addres
```

```
ipDenylistRegex: # [string] IP denylist regex - Messages from matched IP addresse
elasticAPI: # [string] [required] Elasticsearch API endpoint - Absolute path on w
authType: # [string] Authentication type

# -------------- if authType is basic ---------------

username: # [string] Username
password: # [string] Password

# ---------------------------------------------------------


# -------------- if authType is credentialsSecret ---------------

credentialsSecret: # [string] Credentials secret - Select or create a secret that

# ---------------------------------------------------------


# -------------- if authType is authTokens ---------------

authTokens: # [array of strings] Token - Bearer tokens to include in the authoriz

# ---------------------------------------------------------

apiVersion: # [string] API version - The API version to use for communicating wit

# -------------- if apiVersion is custom ---------------

customAPIVersion: # [string] Custom API Version - Custom version information to r

# ---------------------------------------------------------

extraHttpHeaders: # [array] Extra HTTP headers - Headers to add to all events
  - name: # [string] Field Name
    value: # [string] Field Value
metadata: # [array] Fields - Fields to add to events from this input
  - name: # [string] Field Name
    value: # [string] Value - JavaScript expression to compute field's value, enc
proxyMode: # [object]
  enabled: # [boolean] Enable proxy mode - Enable proxying of non-bulk API reques

  # -------------- if enabled is true ---------------
```

```
    url: # [string] Proxy URL - URL of the Elastic server to proxy non-bulk request
    rejectUnauthorized: # [boolean] Reject unauthorized certificates - Reject certi
    removeHeaders: # [array of strings] Remove headers - List of headers to remove
    timeoutSec: # [number; minimum: 1, maximum: 9007199254740991] Proxy request tim
    authType: # [string] Authentication method - Enter credentials directly, or sel

    # ------------------------------------------------------

  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Use QuickConnect - Direct connections to Destinations, and
    - pipeline: # [string] Pipeline or Pack
      output: # [string] Destination

  # ------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a sp
  pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # ------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
confluent_cloud_input: # [object]
  type: # [string] Input Type
  brokers: # [array of strings] Bootstrap servers - List of Confluent Cloud bootstr
  tls: # [object] TLS settings (client side)
    disabled: # [boolean] Disabled
```

```
  # -------------- if disabled is false ---------------

  rejectUnauthorized: # [boolean] Validate server certs - Reject certificates tha
                trusted CA (such as the system's). Defaults to Enabled. Overrides
  servername: # [string] Server name (SNI) - Server name for the SNI (Server Name
  certificateName: # [string] Certificate - The name of the predefined certificat
  caPath: # [string] CA certificate path - Path on client in which to find CA cer
  privKeyPath: # [string] Private key path (mutual auth) - Path on client in whic
  certPath: # [string] Certificate path (mutual auth) - Path on client in which t
  passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
  minVersion: # [string] Minimum TLS version
  maxVersion: # [string] Maximum TLS version

  # -------------------------------------------------------

topics: # [array of strings] [required] Topic - Topic to subscribe to. Warning: T
groupId: # [string] Group ID - The consumer group to which this instance belongs.
fromBeginning: # [boolean] From beginning - Leave enabled if you want the Source,
kafkaSchemaRegistry: # [object] Kafka Schema Registry Authentication
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  schemaRegistryURL: # [string] Schema Registry URL - URL for accessing the Confl
  schemaType: # [string] Schema type - The schema format used to encode and decod
  connectionTimeout: # [number; minimum: 1000, maximum: 60000] Connection timeout
  requestTimeout: # [number; minimum: 1000, maximum: 60000] Request timeout (ms)
  maxRetries: # [number; maximum: 100] Retry limit - Maximum number of times to t
  auth: # [object]  - Credentials to use when authenticating with the schema regi
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    credentialsSecret: # [string] Credentials secret - Select or create a secret

    # -------------------------------------------------------

  tls: # [object] TLS settings (client side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------
```

```
    rejectUnauthorized: # [boolean] Validate server certs - Reject certificates t
                trusted CA (such as the system's). Defaults to Enabled. Overrides
    servername: # [string] Server name (SNI) - Server name for the SNI (Server Na
    certificateName: # [string] Certificate - The name of the predefined certific
    caPath: # [string] CA certificate path - Path on client in which to find CA c
    privKeyPath: # [string] Private key path (mutual auth) - Path on client in wh
    certPath: # [string] Certificate path (mutual auth) - Path on client in which
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
    minVersion: # [string] Minimum TLS version
    maxVersion: # [string] Maximum TLS version

    # ------------------------------------------------------


  # ------------------------------------------------------

connectionTimeout: # [number; minimum: 1000, maximum: 3600000] Connection timeout
requestTimeout: # [number; minimum: 1000, maximum: 3600000] Request timeout (ms)
maxRetries: # [number; maximum: 100] Retry limit - If messages are failing, you c
maxBackOff: # [number; minimum: 30000, maximum: 180000] Backoff limit (ms) - The
initialBackoff: # [number; minimum: 300, maximum: 600000] Initial retry interval
backoffRate: # [number; minimum: 2, maximum: 20] Backoff multiplier - Set the bac
authenticationTimeout: # [number; minimum: 1000, maximum: 3600000] Authentication
reauthenticationThreshold: # [number; minimum: 1000, maximum: 1800000] Reauthenti
sasl: # [object] Authentication - Authentication parameters to use when connectin
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  mechanism: # [string] SASL mechanism
  oauthEnabled: # [boolean] Enable OAuth - Enable OAuth authentication

  # ------------------------------------------------------

sessionTimeout: # [number; minimum: 1000, maximum: 3600000] Session timeout (ms)
  Timeout used to detect client failures when using Kafka's group-management faci
  If the client sends no heartbeats to the broker before the timeout expires,
  the broker will remove the client from the group and initiate a rebalance.
  Value must be between the broker's configured group.min.session.timeout.ms and
  See [Kafka's documentation](https://kafka.apache.org/documentation/#consumercor
rebalanceTimeout: # [number; minimum: 1000, maximum: 3600000] Rebalance timeout (
  Maximum allowed time for each worker to join the group after a rebalance begins
  If the timeout is exceeded, the coordinator broker will remove the worker from
```

```
  See [Kafka's documentation](https://kafka.apache.org/documentation/#connectconf
heartbeatInterval: # [number; minimum: 1000, maximum: 3600000] Heartbeat interval
  Expected time between heartbeats to the consumer coordinator when using Kafka's
  Value must be lower than sessionTimeout and typically should not exceed 1/3 of
  See [Kafka's documentation](https://kafka.apache.org/documentation/#consumercon
autoCommitInterval: # [number; minimum: 1000, maximum: 3600000] Offset commit int
autoCommitThreshold: # [number; minimum: 1, maximum: 10000] Offset commit thresho
maxBytesPerPartition: # [number; minimum: 1, maximum: 10000000] Byte limit, per p
maxBytes: # [number; minimum: 1, maximum: 1000000000] Byte limit - Maximum number
maxSocketErrors: # [number; maximum: 100] Error limit, per socket - Maximum numbe
metadata: # [array] Fields - Fields to add to events from this input
  - name: # [string] Field Name
    value: # [string] Value - JavaScript expression to compute field's value, enc
description: # [string] Description
disabled: # [boolean] Disabled
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

# -------------- if sendToRoutes is false ---------------

connections: # [array] Use QuickConnect - Direct connections to Destinations, and
  - pipeline: # [string] Pipeline or Pack
    output: # [string] Destination

# --------------------------------------------------------

environment: # [string] Environment - Optionally, enable this config only on a sp
pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

# -------------- if pqEnabled is true ---------------

pq: # [object]
  mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
  maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
  commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
  maxFileSize: # [string] File size limit - The maximum size to store in each que
  maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
  path: # [string] Queue file path - The location for the persistent queue files.
  compress: # [string] Compression - Codec to use to compress the persisted data

# --------------------------------------------------------

streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
```

```
grafana_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled
  host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses
  port: # [number; maximum: 65535] [required] Port - Port to listen on
  tls: # [object] TLS settings (server side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    certificateName: # [string] Certificate - The name of the predefined certificat
    privKeyPath: # [string] Private key path - Path on server containing the privat
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
    certPath: # [string] Certificate path - Path on server containing certificates
    caPath: # [string] CA certificate path - Path on server containing CA certifica
    requestCert: # [boolean] Authenticate client (mutual auth) - Require clients to
    minVersion: # [string] Minimum TLS version
    maxVersion: # [string] Maximum TLS version

    # ----------------------------------------------------

  maxActiveReq: # [number] Active request limit - Maximum number of active requests
  maxRequestsPerSocket: # [integer] Requests-per-socket limit - Maximum number of r
  enableProxyHeader: # [boolean] Show originating IP - Extract the client IP and po
  captureHeaders: # [boolean] Capture request headers - Add request headers to ever
  activityLogSampleRate: # [number; minimum: 1] Activity log sample rate - How ofte
  requestTimeout: # [number] Request timeout (seconds) - How long to wait for an in
  socketTimeout: # [number] Socket timeout (seconds) - How long Cribl Edge should w
  keepAliveTimeout: # [number; minimum: 1, maximum: 600] Keep alive timeout (second
  enableHealthCheck: # [boolean] Health check endpoint - Expose the /cribl_health e
  ipAllowlistRegex: # [string] IP allowlist regex - Messages from matched IP addres
  ipDenylistRegex: # [string] IP denylist regex - Messages from matched IP addresse
  prometheusAPI: # [string] Remote Write API endpoint - Absolute path on which to l
  lokiAPI: # [string] Logs API endpoint - Absolute path on which to listen for Loki
  prometheusAuth: # [object]
    authType: # [string] Authentication type - Remote Write authentication type

    # -------------- if authType is basic ---------------

    username: # [string] Username
    password: # [string] Password

    # ----------------------------------------------------
```

```
    # -------------- if authType is token ---------------

    token: # [string] Token - Bearer token to include in the authorization header

    # ----------------------------------------------------


    # -------------- if authType is credentialsSecret ---------------

    credentialsSecret: # [string] Credentials secret - Select or create a secret th

    # ----------------------------------------------------


    # -------------- if authType is textSecret ---------------

    textSecret: # [string] Token (text secret) - Select or create a stored text sec

    # ----------------------------------------------------


    # -------------- if authType is oauth ---------------

    loginUrl: # [string] Login URL - URL for OAuth
    secretParamName: # [string] OAuth Secret parameter name - Secret parameter name
    secret: # [string] OAuth secret - Secret parameter value to pass in request boc
    tokenAttributeName: # [string] Token attribute name - Name of the auth token at
    spacer: # [null]
    authHeaderExpr: # [string] Authorize expression - JavaScript expression to comp
    tokenTimeoutSecs: # [number; minimum: 1, maximum: 300000] Refresh interval (sec
    oauthParams: # [array] OAuth parameters - Additional parameters to send in the
      - name: # [string] Name - OAuth parameter name
        value: # [string] Value - OAuth parameter value
    oauthHeaders: # [array] OAuth headers - Additional headers to send in the OAuth
      - name: # [string] Name - OAuth header name
        value: # [string] Value - OAuth header value

    # ----------------------------------------------------

  lokiAuth: # [object]
    authType: # [string] Authentication type - Loki logs authentication type
```

```
# -------------- if authType is basic ---------------

username: # [string] Username
password: # [string] Password

# ----------------------------------------------------


# -------------- if authType is token ---------------

token: # [string] Token - Bearer token to include in the authorization header

# ----------------------------------------------------


# -------------- if authType is credentialsSecret ---------------

credentialsSecret: # [string] Credentials secret - Select or create a secret th

# ----------------------------------------------------


# -------------- if authType is textSecret ---------------

textSecret: # [string] Token (text secret) - Select or create a stored text sec

# ----------------------------------------------------


# -------------- if authType is oauth ---------------

loginUrl: # [string] Login URL - URL for OAuth
secretParamName: # [string] OAuth Secret parameter name - Secret parameter name
secret: # [string] OAuth secret - Secret parameter value to pass in request bod
tokenAttributeName: # [string] Token attribute name - Name of the auth token at
spacer: # [null]
authHeaderExpr: # [string] Authorize expression - JavaScript expression to comp
tokenTimeoutSecs: # [number; minimum: 1, maximum: 300000] Refresh interval (sec
oauthParams: # [array] OAuth parameters - Additional parameters to send in the
  - name: # [string] Name - OAuth parameter name
    value: # [string] Value - OAuth parameter value
oauthHeaders: # [array] OAuth headers - Additional headers to send in the OAuth
```

```
      - name: # [string] Name - OAuth header name
        value: # [string] Value - OAuth header value

    # -------------------------------------------------------

  metadata: # [array] Fields - Fields to add to events from this input
    - name: # [string] Field Name
      value: # [string] Value - JavaScript expression to compute field's value, enc
  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Use QuickConnect - Direct connections to Destinations, and
    - pipeline: # [string] Pipeline or Pack
      output: # [string] Destination

  # -------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a sp
  pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # -------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
loki_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled
  host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses
  port: # [number; maximum: 65535] [required] Port - Port to listen on
  tls: # [object] TLS settings (server side)
```

```
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  certificateName: # [string] Certificate - The name of the predefined certificat
  privKeyPath: # [string] Private key path - Path on server containing the privat
  passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
  certPath: # [string] Certificate path - Path on server containing certificates
  caPath: # [string] CA certificate path - Path on server containing CA certifica
  requestCert: # [boolean] Authenticate client (mutual auth) - Require clients to
  minVersion: # [string] Minimum TLS version
  maxVersion: # [string] Maximum TLS version

  # -------------------------------------------------------

maxActiveReq: # [number] Active request limit - Maximum number of active requests
maxRequestsPerSocket: # [integer] Requests-per-socket limit - Maximum number of r
enableProxyHeader: # [boolean] Show originating IP - Extract the client IP and po
captureHeaders: # [boolean] Capture request headers - Add request headers to ever
activityLogSampleRate: # [number; minimum: 1] Activity log sample rate - How ofte
requestTimeout: # [number] Request timeout (seconds) - How long to wait for an ir
socketTimeout: # [number] Socket timeout (seconds) - How long Cribl Edge should w
keepAliveTimeout: # [number; minimum: 1, maximum: 600] Keep-alive timeout (second
enableHealthCheck: # [boolean] Health check endpoint - Expose the /cribl_health e
ipAllowlistRegex: # [string] IP allowlist regex - Messages from matched IP addres
ipDenylistRegex: # [string] IP denylist regex - Messages from matched IP addresse
lokiAPI: # [string] [required] Logs API endpoint - Absolute path on which to list
authType: # [string] Authentication type - Loki logs authentication type

# -------------- if authType is basic ---------------

username: # [string] Username
password: # [string] Password

# -------------------------------------------------------


# -------------- if authType is token ---------------

token: # [string] Token - Bearer token to include in the authorization header

# -------------------------------------------------------
```

```
# -------------- if authType is credentialsSecret ---------------

credentialsSecret: # [string] Credentials secret - Select or create a secret that

# ----------------------------------------------------------


# -------------- if authType is textSecret ---------------

textSecret: # [string] Token (text secret) - Select or create a stored text secre

# ----------------------------------------------------------


# -------------- if authType is oauth ---------------

loginUrl: # [string] Login URL - URL for OAuth
secretParamName: # [string] OAuth Secret parameter name - Secret parameter name t
secret: # [string] OAuth secret - Secret parameter value to pass in request body
tokenAttributeName: # [string] Token attribute name - Name of the auth token attr
spacer: # [null]
authHeaderExpr: # [string] Authorize expression - JavaScript expression to comput
tokenTimeoutSecs: # [number; minimum: 1, maximum: 300000] Refresh interval (secs.
oauthParams: # [array] OAuth parameters - Additional parameters to send in the OA
  - name: # [string] Name - OAuth parameter name
    value: # [string] Value - OAuth parameter value
oauthHeaders: # [array] OAuth headers - Additional headers to send in the OAuth t
  - name: # [string] Name - OAuth header name
    value: # [string] Value - OAuth header value

# ----------------------------------------------------------

metadata: # [array] Fields - Fields to add to events from this input
  - name: # [string] Field Name
    value: # [string] Value - JavaScript expression to compute field's value, enc
description: # [string] Description
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

# -------------- if sendToRoutes is false ---------------

connections: # [array] Use QuickConnect - Direct connections to Destinations, and
```

```
    - pipeline: # [string] Pipeline or Pack
      output: # [string] Destination

  # ------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a sp
  pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # ------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
prometheus_rw_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled
  host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses
  port: # [number; maximum: 65535] [required] Port - Port to listen on
  tls: # [object] TLS settings (server side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    certificateName: # [string] Certificate - The name of the predefined certificat
    privKeyPath: # [string] Private key path - Path on server containing the privat
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
    certPath: # [string] Certificate path - Path on server containing certificates
    caPath: # [string] CA certificate path - Path on server containing CA certifica
    requestCert: # [boolean] Authenticate client (mutual auth) - Require clients to
    minVersion: # [string] Minimum TLS version
    maxVersion: # [string] Maximum TLS version

    # ------------------------------------------------------
```

```
maxActiveReq: # [number] Active request limit - Maximum number of active requests
maxRequestsPerSocket: # [integer] Requests-per-socket limit - Maximum number of r
enableProxyHeader: # [boolean] Show originating IP - Extract the client IP and po
captureHeaders: # [boolean] Capture request headers - Add request headers to ever
activityLogSampleRate: # [number; minimum: 1] Activity log sample rate - How ofte
requestTimeout: # [number] Request timeout (seconds) - How long to wait for an ir
socketTimeout: # [number] Socket timeout (seconds) - How long Cribl Edge should w
keepAliveTimeout: # [number; minimum: 1, maximum: 600] Keep-alive timeout (second
enableHealthCheck: # [boolean] Health check endpoint - Expose the /cribl_health e
ipAllowlistRegex: # [string] IP allowlist regex - Messages from matched IP addres
ipDenylistRegex: # [string] IP denylist regex - Messages from matched IP addresse
prometheusAPI: # [string] [required] Remote Write API endpoint - Absolute path or
authType: # [string] Authentication type - Remote Write authentication type

# -------------- if authType is basic ---------------

username: # [string] Username
password: # [string] Password

# ---------------------------------------------------------


# -------------- if authType is token ---------------

token: # [string] Token - Bearer token to include in the authorization header

# ---------------------------------------------------------


# -------------- if authType is credentialsSecret ---------------

credentialsSecret: # [string] Credentials secret - Select or create a secret that

# ---------------------------------------------------------


# -------------- if authType is textSecret ---------------

textSecret: # [string] Token (text secret) - Select or create a stored text secre

# ---------------------------------------------------------
```

```
# -------------- if authType is oauth ---------------

loginUrl: # [string] Login URL - URL for OAuth
secretParamName: # [string] OAuth Secret parameter name - Secret parameter name t
secret: # [string] OAuth secret - Secret parameter value to pass in request body
tokenAttributeName: # [string] Token attribute name - Name of the auth token attr
spacer: # [null]
authHeaderExpr: # [string] Authorize expression - JavaScript expression to comput
tokenTimeoutSecs: # [number; minimum: 1, maximum: 300000] Refresh interval (secs.
oauthParams: # [array] OAuth parameters - Additional parameters to send in the OA
  - name: # [string] Name - OAuth parameter name
    value: # [string] Value - OAuth parameter value
oauthHeaders: # [array] OAuth headers - Additional headers to send in the OAuth l
  - name: # [string] Name - OAuth header name
    value: # [string] Value - OAuth header value

# --------------------------------------------------------

metadata: # [array] Fields - Fields to add to events from this input
  - name: # [string] Field Name
    value: # [string] Value - JavaScript expression to compute field's value, enc
description: # [string] Description
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

# -------------- if sendToRoutes is false ---------------

connections: # [array] Use QuickConnect - Direct connections to Destinations, and
  - pipeline: # [string] Pipeline or Pack
    output: # [string] Destination

# --------------------------------------------------------

environment: # [string] Environment - Optionally, enable this config only on a sp
pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

# -------------- if pqEnabled is true ---------------

pq: # [object]
  mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
  maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
  commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
  maxFileSize: # [string] File size limit - The maximum size to store in each que
```

```
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # ------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
prometheus_input: # [object]
  type: # [string] Input Type
  dimensionList: # [array of strings] Extra Dimensions - Other dimensions to includ
  discoveryType: # [string] Discovery Type - Target discovery mechanism. Use static

  # -------------- if discoveryType is static ---------------

  targetList: # [array of strings] Targets - List of Prometheus targets to pull met

  # ------------------------------------------------------


  # -------------- if discoveryType is dns ---------------

  nameList: # [array of strings] DNS Names - List of DNS names to resolve
  recordType: # [string] Record Type - DNS Record type to resolve
  scrapeProtocol: # [string] Metrics Protocol - Protocol to use when collecting met
  scrapePath: # [string] Metrics Path - Path to use when collecting metrics from di

  # ------------------------------------------------------


  # -------------- if discoveryType is ec2 ---------------

  usePublicIp: # [boolean] Use Public IP - Use public IP address for discovered tar
  scrapeProtocol: # [string] Metrics Protocol - Protocol to use when collecting met
  scrapePort: # [number; minimum: 1, maximum: 65535] Metrics Port - The port number
  scrapePath: # [string] Metrics Path - Path to use when collecting metrics from di
  searchFilter: # [array] Search Filter - EC2 Instance Search Filter
    - Name: # [string] Filter Name - Search filter attribute name, see: https://doc
      Values: # [array of strings] Filter Values - Search Filter Values, if empty d
  awsAuthenticationMethod: # [string] Authentication method - AWS authentication me
  awsSecretKey: # [string] Secret key
  region: # [string] Region - Region where the EC2 is located
  endpoint: # [string] Endpoint - EC2 service endpoint. If empty, defaults to the A
  signatureVersion: # [string] Signature version - Signature version to use for sig
```

```
reuseConnections: # [boolean] Reuse connections - Reuse connections between reque
rejectUnauthorized: # [boolean] Reject unauthorized certificates - Reject certifi
enableAssumeRole: # [boolean] Enable for EC2 - Use Assume Role credentials to acc
assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the role
assumeRoleExternalId: # [string] External ID - External ID to use when assuming r
durationSeconds: # [number; minimum: 900, maximum: 43200] Duration (seconds) - Du

# -------------------------------------------------------

interval: # [number; minimum: 1, maximum: 60] Poll Interval - How often in minute
logLevel: # [string] [required] Log Level - Collector runtime Log Level
rejectUnauthorized: # [boolean] Reject unauthorized certificates - Reject certifi
keepAliveTime: # [number; minimum: 10] Keep alive time (seconds) - How often work
jobTimeout: # [string] Job timeout - Maximum time the job is allowed to run (e.g.
maxMissedKeepAlives: # [number; minimum: 2] Worker timeout (periods) - The number
ttl: # [string] Time to live - Time to keep the job's artifacts on disk after job
ignoreGroupJobsLimit: # [boolean] Ignore Worker Group job limits - When enabled,
metadata: # [array] Fields - Fields to add to events from this input
  - name: # [string] Field Name
    value: # [string] Value - JavaScript expression to compute field's value, enc
authType: # [string] Authentication method - Enter credentials directly, or selec
username: # [string] Username - Username for Prometheus Basic authentication
password: # [string] Password - Password for Prometheus Basic authentication

# -------------- if authType is manual ---------------


# -------------------------------------------------------

credentialsSecret: # [string] Credentials secret - Select or create a secret that

# -------------- if authType is secret ---------------


# -------------------------------------------------------

description: # [string] Description
disabled: # [boolean] Disabled
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

# -------------- if sendToRoutes is false ---------------
```

```
  connections: # [array] Use QuickConnect - Direct connections to Destinations, and
    - pipeline: # [string] Pipeline or Pack
      output: # [string] Destination

  # ------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a sp
  pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # ------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
edge_prometheus_input: # [object]
  type: # [string] Input Type
  dimensionList: # [array of strings] Extra Dimensions - Other dimensions to includ
  discoveryType: # [string] [required] Discovery Type - Target discovery mechanism.

  # -------------- if discoveryType is static ---------------

  targets: # [array] Targets
    - protocol: # [string] Protocol - Protocol to use when collecting metrics
      host: # [string] Host - Name of host from which to pull metrics.
      port: # [number; minimum: 1, maximum: 65535] Port - The port number in the me
      path: # [string] Path - Path to use when collecting metrics from discovered t

  # ------------------------------------------------------



  # -------------- if discoveryType is dns ---------------

  nameList: # [array of strings] DNS Names - List of DNS names to resolve
  recordType: # [string] Record Type - DNS Record type to resolve
```

```
scrapeProtocol: # [string] Metrics Protocol - Protocol to use when collecting met
scrapePath: # [string] Metrics Path - Path to use when collecting metrics from di

# ------------------------------------------------------


# -------------- if discoveryType is ec2 ---------------

usePublicIp: # [boolean] Use Public IP - Use public IP address for discovered tar
scrapeProtocol: # [string] Metrics Protocol - Protocol to use when collecting met
scrapePort: # [number; minimum: 1, maximum: 65535] Metrics Port - The port numbe
scrapePath: # [string] Metrics Path - Path to use when collecting metrics from di
searchFilter: # [array] Search Filter - EC2 Instance Search Filter
  - Name: # [string] Filter Name - Search filter attribute name, see: https://doc
    Values: # [array of strings] Filter Values - Search Filter Values, if empty c
awsAuthenticationMethod: # [string] Authentication method - AWS authentication me
awsSecretKey: # [string] Secret key
region: # [string] Region - Region where the EC2 is located
endpoint: # [string] Endpoint - EC2 service endpoint. If empty, defaults to the A
signatureVersion: # [string] Signature version - Signature version to use for sig
reuseConnections: # [boolean] Reuse connections - Reuse connections between reque
rejectUnauthorized: # [boolean] Reject unauthorized certificates - Reject certifi
enableAssumeRole: # [boolean] Enable for EC2 - Use Assume Role credentials to acc
assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the role
assumeRoleExternalId: # [string] External ID - External ID to use when assuming r
durationSeconds: # [number; minimum: 900, maximum: 43200] Duration (seconds) - Du

# ------------------------------------------------------


# -------------- if discoveryType is k8s-node ---------------

scrapeProtocol: # [string] Protocol - Protocol to use when collecting metrics
scrapePort: # [number; minimum: 1, maximum: 65535] Port - The port number in the
scrapePath: # [string] Path - Path to use when collecting metrics from discovered

# ------------------------------------------------------


# -------------- if discoveryType is k8s-pods ---------------

scrapeProtocolExpr: # [string] Protocol - Protocol to use when collecting metrics
scrapePortExpr: # [string] Port - The port number in the metrics URL for discover
```

```
  scrapePathExpr: # [string] Path - Path to use when collecting metrics from discov
  podFilter: # [array] Filter Rules -
Add rules to decide which pods to discover for metrics.
Pods are searched if no rules are given or of all the rules'
expressions evaluate to true.

    - filter: # [string] Filter Expression - JavaScript expression applied to pods
      description: # [string] Description - Optional description of this rule's pur

  # ------------------------------------------------------

  interval: # [number; minimum: 2] Poll Interval - How often in seconds to scrape t
  timeout: # [number; maximum: 60000] HTTP Connection Timeout - Timeout, in millise
  persistence: # [object] Disk Spooling
    enable: # [boolean] Enable disk spooling - Spool events on disk for Cribl Edge

    # -------------- if enable is true ---------------

    timeWindow: # [string] Bucket time span - Time period for grouping spooled ever
    maxDataSize: # [string] Data size limit - Maximum disk space that can be consum
    maxDataTime: # [string] Data age limit - Maximum amount of time to retain data
    compress: # [string] Compression - Data compression format. Default is gzip.

    # ------------------------------------------------------

  metadata: # [array] Fields - Fields to add to events from this input
    - name: # [string] Field Name
      value: # [string] Value - JavaScript expression to compute field's value, enc
  authType: # [string] Authentication method - Enter credentials directly, or selec
  username: # [string] Username - Username for Prometheus Basic authentication
  password: # [string] Password - Password for Prometheus Basic authentication

  # -------------- if authType is manual ---------------


  # ------------------------------------------------------

  credentialsSecret: # [string] Credentials secret - Select or create a secret that

  # -------------- if authType is secret ---------------


  # ------------------------------------------------------
```

```
  description: # [string] Description
  disabled: # [boolean] Disabled
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Use QuickConnect - Direct connections to Destinations, and
    - pipeline: # [string] Pipeline or Pack
      output: # [string] Destination

  # --------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a sp
  pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # --------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
office365_mgmt_input: # [object]
  type: # [string] Input Type
  planType: # [string] [required] Subscription plan - Office 365 subscription plan
  tenantId: # [string] Tenant ID - Office 365 Azure Tenant ID
  appId: # [string] [required] App ID - Office 365 Azure Application ID
  timeout: # [number; maximum: 2400] Request timeout (seconds) - HTTP request inact
  keepAliveTime: # [number; minimum: 10] Keep alive time (seconds) - How often work
  jobTimeout: # [string] Job timeout - Maximum time the job is allowed to run (e.g.
  maxMissedKeepAlives: # [number; minimum: 2] Worker timeout (periods) - The number
  ttl: # [string] Time to live - Time to keep the job's artifacts on disk after job
  ignoreGroupJobsLimit: # [boolean] Ignore Worker Group job limits - When enabled,
  metadata: # [array] Fields - Fields to add to events from this input
```

```
  - name: # [string] Field Name
    value: # [string] Value - JavaScript expression to compute field's value, enc
publisherIdentifier: # [string] Publisher Identifier - Optional Publisher Identif
contentConfig: # [array] Content Types - Enable Office 365 Management Activity AP
  - contentType: # [string] Content Type - Office 365 Management Activity API Cor
    description: # [string] Interval Description - If interval type is minutes th
    interval: # [number; minimum: 1, maximum: 60] Interval
    logLevel: # [string] Log Level - Collector runtime Log Level
    enabled: # [boolean] Enabled
ingestionLag: # [number; maximum: 7200] Ingestion lag (minutes) - Use this settir
retryRules: # [object]
  type: # [string] Retry type - The algorithm to use when performing HTTP retries

  # -------------- if type is static ---------------

  codes: # [array of numbers] Retry HTTP codes - List of http codes that trigger
  interval: # [number; maximum: 20000] Wait (ms) - Time interval between failed r
  limit: # [number; maximum: 20] Retry limit - The maximum number of times to ret
  enableHeader: # [boolean] Honor Retry-After header - Honor any Retry-After head
  retryConnectTimeout: # [boolean] Retry connection timeout - Make a single retry
  retryConnectReset: # [boolean] Retry connection reset - Retry request when a co

  # -------------------------------------------------------


  # -------------- if type is backoff ---------------

  interval: # [number; maximum: 20000] Initial retry interval (ms) - Time interva
  limit: # [number; maximum: 20] Retry limit - The maximum number of times to ret
  multiplier: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base for e
  codes: # [array of numbers] Retry HTTP codes - List of http codes that trigger
  enableHeader: # [boolean] Honor Retry-After header - Honor any Retry-After head
  retryConnectTimeout: # [boolean] Retry connection timeout - Make a single retry
  retryConnectReset: # [boolean] Retry connection reset - Retry request when a co

  # -------------------------------------------------------

authType: # [string] Authentication method - Enter client secret directly, or sel
clientSecret: # [string] Client secret - Office 365 Azure client secret

# -------------- if authType is manual ---------------
```

```
  # ------------------------------------------------------

  textSecret: # [string] Client secret (text secret) - Select or create a stored te

  # -------------- if authType is secret ---------------


  # ------------------------------------------------------

  description: # [string] Description
  disabled: # [boolean] Disabled
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Use QuickConnect - Direct connections to Destinations, and
    - pipeline: # [string] Pipeline or Pack
      output: # [string] Destination

  # ------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a sp
  pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # ------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
office365_service_input: # [object]
  type: # [string] Input Type
  planType: # [string] Subscription plan - Office 365 subscription plan for your or
  tenantId: # [string] Tenant ID - Office 365 Azure Tenant ID
```

```
appId: # [string] [required] App ID - Office 365 Azure Application ID
timeout: # [number; maximum: 2400] Request timeout (seconds) - HTTP request inact
keepAliveTime: # [number; minimum: 10] Keep alive time (seconds) - How often work
jobTimeout: # [string] Job timeout - Maximum time the job is allowed to run (e.g.
maxMissedKeepAlives: # [number; minimum: 2] Worker timeout (periods) - The numbe
ttl: # [string] Time to live - Time to keep the job's artifacts on disk after jo
ignoreGroupJobsLimit: # [boolean] Ignore Worker Group job limits - When enabled,
metadata: # [array] Fields - Fields to add to events from this input
  - name: # [string] Field Name
    value: # [string] Value - JavaScript expression to compute field's value, enc
contentConfig: # [array] Content Types - Enable Office 365 Service Communication
  - contentType: # [string] Content Type - Office 365 Services API Content Type
    description: # [string] Interval Description - If interval type is minutes th
    interval: # [number; maximum: 60] Interval
    logLevel: # [string] Log Level - Collector runtime Log Level
    enabled: # [boolean] Enabled
retryRules: # [object]
  type: # [string] Retry type - The algorithm to use when performing HTTP retries

  # -------------- if type is static ---------------

  codes: # [array of numbers] Retry HTTP codes - List of http codes that trigger
  interval: # [number; maximum: 20000] Wait (ms) - Time interval between failed r
  limit: # [number; maximum: 20] Retry limit - The maximum number of times to ret
  enableHeader: # [boolean] Honor Retry-After header - Honor any Retry-After head
  retryConnectTimeout: # [boolean] Retry connection timeout - Make a single retry
  retryConnectReset: # [boolean] Retry connection reset - Retry request when a co

  # -------------------------------------------------------


  # -------------- if type is backoff ---------------

  interval: # [number; maximum: 20000] Initial retry interval (ms) - Time interva
  limit: # [number; maximum: 20] Retry limit - The maximum number of times to ret
  multiplier: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base for e
  codes: # [array of numbers] Retry HTTP codes - List of http codes that trigger
  enableHeader: # [boolean] Honor Retry-After header - Honor any Retry-After head
  retryConnectTimeout: # [boolean] Retry connection timeout - Make a single retry
  retryConnectReset: # [boolean] Retry connection reset - Retry request when a co

  # -------------------------------------------------------
```

```
authType: # [string] Authentication method - Enter client secret directly, or sel
clientSecret: # [string] Client secret - Office 365 Azure client secret

# -------------- if authType is manual ---------------


# -----------------------------------------------------

textSecret: # [string] Client secret (text secret) - Select or create a stored te

# -------------- if authType is secret ---------------


# -----------------------------------------------------

description: # [string] Description
disabled: # [boolean] Disabled
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

# -------------- if sendToRoutes is false ---------------

connections: # [array] Use QuickConnect - Direct connections to Destinations, and
  - pipeline: # [string] Pipeline or Pack
    output: # [string] Destination

# -----------------------------------------------------

environment: # [string] Environment - Optionally, enable this config only on a sp
pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

# -------------- if pqEnabled is true ---------------

pq: # [object]
  mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
  maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
  commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
  maxFileSize: # [string] File size limit - The maximum size to store in each que
  maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
  path: # [string] Queue file path - The location for the persistent queue files.
  compress: # [string] Compression - Codec to use to compress the persisted data

# -----------------------------------------------------
```

```
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
office365_msg_trace_input: # [object]
  type: # [string] Input Type
  url: # [string] Report URL - URL to use when retrieving report data.
  interval: # [number; minimum: 1, maximum: 60] [required] Poll interval - How ofte
  startDate: # [string] Date range start - Backward offset for the search range's h
  endDate: # [string] Date range end - Backward offset for the search range's tail.
  timeout: # [number; maximum: 2400] Request timeout (seconds) - HTTP request inact
  disableTimeFilter: # [boolean] Disable time filter - Disables time filtering of e
  authType: # [string] Authentication method - Select authentication method.

  # -------------- if authType is manual ---------------

  username: # [string] Username - Username to run Message Trace API call.
  password: # [string] Password - Password to run Message Trace API call.

  # ----------------------------------------------------


  # -------------- if authType is secret ---------------

  credentialsSecret: # [string] Credentials secret - Select or create a secret that

  # ----------------------------------------------------


  # -------------- if authType is oauth ---------------

  clientSecret: # [string] Client secret - client_secret to pass in the OAuth reque
  tenantId: # [string] Tenant identifier - Directory ID (tenant identifier) in Azur
  clientId: # [string] Client ID - client_id to pass in the OAuth request parameter
  resource: # [string] Resource - Resource to pass in the OAuth request parameter.
  planType: # [string] Subscription plan - Office 365 subscription plan for your or

  # ----------------------------------------------------


  # -------------- if authType is oauthSecret ---------------

  textSecret: # [string] Client secret - Select or create a secret that references
  tenantId: # [string] Tenant identifier - Directory ID (tenant identifier) in Azur
  clientId: # [string] Client ID - client_id to pass in the OAuth request parameter
```

```
resource: # [string] Resource - Resource to pass in the OAuth request parameter.
planType: # [string] Subscription plan - Office 365 subscription plan for your or

# ------------------------------------------------------


# -------------- if authType is oauthCert ---------------

certOptions: # [object]
  certificateName: # [string] Certificate - The name of the predefined certificat
  privKeyPath: # [string] Private key path - Path to the private key to use. Key
  passphrase: # [string] Passphrase - Passphrase to use to decrypt the private ke
  certPath: # [string] [required] Certificate path - Path to the certificate to u
tenantId: # [string] Tenant identifier - Directory ID (tenant identifier) in Azur
clientId: # [string] Client ID - client_id to pass in the OAuth request parameter
resource: # [string] Resource - Resource to pass in the OAuth request parameter.
planType: # [string] Subscription plan - Office 365 subscription plan for your or

# ------------------------------------------------------

rescheduleDroppedTasks: # [boolean] Reschedule tasks - Reschedule tasks that fail
maxTaskReschedule: # [number; minimum: 1] Task reschedule limit - Maximum number
logLevel: # [string] Log level - Log Level (verbosity) for collection runtime beh
jobTimeout: # [string] Job timeout - Maximum time the job is allowed to run (e.g.
keepAliveTime: # [number; minimum: 10] Keep alive time (seconds) - How often work
maxMissedKeepAlives: # [number; minimum: 2] Worker timeout (periods) - The number
ttl: # [string] Time to live - Time to keep the job's artifacts on disk after job
ignoreGroupJobsLimit: # [boolean] Ignore Worker Group job limits - When enabled,
metadata: # [array] Fields - Fields to add to events from this input
  - name: # [string] Field Name
    value: # [string] Value - JavaScript expression to compute field's value, enc
retryRules: # [object]
  type: # [string] Retry type - The algorithm to use when performing HTTP retries

  # -------------- if type is static ---------------

  codes: # [array of numbers] Retry HTTP codes - List of http codes that trigger
  interval: # [number; maximum: 20000] Wait (ms) - Time interval between failed r
  limit: # [number; maximum: 20] Retry limit - The maximum number of times to ret
  enableHeader: # [boolean] Honor Retry-After header - Honor any Retry-After head
  retryConnectTimeout: # [boolean] Retry connection timeout - Make a single retry
  retryConnectReset: # [boolean] Retry connection reset - Retry request when a co
```

```
  # ------------------------------------------------------


  # -------------- if type is backoff ---------------

  interval: # [number; maximum: 20000] Initial retry interval (ms) - Time interva
  limit: # [number; maximum: 20] Retry limit - The maximum number of times to ret
  multiplier: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base for e
  codes: # [array of numbers] Retry HTTP codes - List of http codes that trigger
  enableHeader: # [boolean] Honor Retry-After header - Honor any Retry-After head
  retryConnectTimeout: # [boolean] Retry connection timeout - Make a single retry
  retryConnectReset: # [boolean] Retry connection reset - Retry request when a co

  # ------------------------------------------------------

description: # [string] Description
disabled: # [boolean] Disabled
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

# -------------- if sendToRoutes is false ---------------

connections: # [array] Use QuickConnect - Direct connections to Destinations, and
  - pipeline: # [string] Pipeline or Pack
    output: # [string] Destination

# ------------------------------------------------------

environment: # [string] Environment - Optionally, enable this config only on a sp
pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

# -------------- if pqEnabled is true ---------------

pq: # [object]
  mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
  maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
  commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
  maxFileSize: # [string] File size limit - The maximum size to store in each que
  maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
  path: # [string] Queue file path - The location for the persistent queue files.
  compress: # [string] Compression - Codec to use to compress the persisted data

# ------------------------------------------------------
```

```
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
eventhub_input: # [object]
  type: # [string] Input Type
  brokers: # [array of strings] Brokers - List of Event Hubs Kafka brokers to conne
  topics: # [array of strings] [required] Event Hub name - The name of the Event Hu
  groupId: # [string] Group ID - The consumer group this instance belongs to. Defau
  fromBeginning: # [boolean] From beginning - Start reading from earliest available
  connectionTimeout: # [number; minimum: 1000, maximum: 3600000] Connection timeout
  requestTimeout: # [number; minimum: 1000, maximum: 3600000] Request timeout (ms)
  maxRetries: # [number; maximum: 100] Retry limit - If messages are failing, you c
  maxBackOff: # [number; minimum: 30000, maximum: 180000] Backoff limit (ms) - The
  initialBackoff: # [number; minimum: 300, maximum: 600000] Initial retry interval
  backoffRate: # [number; minimum: 2, maximum: 20] Backoff multiplier - Set the bac
  authenticationTimeout: # [number; minimum: 1000, maximum: 3600000] Authentication
  reauthenticationThreshold: # [number; minimum: 1000, maximum: 1800000] Reauthenti
  sasl: # [object] Authentication - Authentication parameters to use when connecti
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    mechanism: # [string] SASL mechanism

    # ----------------------------------------------------

  tls: # [object] TLS settings (client side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    rejectUnauthorized: # [boolean] Validate server certs - Reject certificates tha

    # ----------------------------------------------------

  sessionTimeout: # [number; minimum: 6000, maximum: 300000] Session timeout (ms) -
    Timeout (session.timeout.ms in Kafka domain) used to detect client failures whe
    If the client sends no heartbeats to the broker before the timeout expires, the
    Value must be lower than rebalanceTimeout.
    See details [here](https://github.com/Azure/azure-event-hubs-for-kafka/blob/mas
  rebalanceTimeout: # [number; minimum: 1000, maximum: 3600000] Rebalance timeout (
    Maximum allowed time (rebalance.timeout.ms in Kafka domain) for each worker to
    If the timeout is exceeded, the coordinator broker will remove the worker from
    See [Recommended configurations](https://github.com/Azure/azure-event-hubs-for-
```

```
heartbeatInterval: # [number; minimum: 1000, maximum: 3600000] Heartbeat interval
  Expected time (heartbeat.interval.ms in Kafka domain) between heartbeats to the
  Value must be lower than sessionTimeout and typically should not exceed 1/3 of
  See [Recommended configurations](https://github.com/Azure/azure-event-hubs-for-
autoCommitInterval: # [number; minimum: 1000, maximum: 3600000] Offset commit int
autoCommitThreshold: # [number; minimum: 1, maximum: 10000] Offset commit thresho
maxBytesPerPartition: # [number; minimum: 1, maximum: 10000000] Byte limit, per p
maxBytes: # [number; minimum: 1, maximum: 1000000000] Byte limit - Maximum number
maxSocketErrors: # [number; maximum: 100] Error limit, per socket - Maximum numbe
minimizeDuplicates: # [boolean] Minimize duplicates - Minimize duplicate events b
metadata: # [array] Fields - Fields to add to events from this input
  - name: # [string] Field Name
    value: # [string] Value - JavaScript expression to compute field's value, enc
description: # [string] Description
disabled: # [boolean] Disabled
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

# -------------- if sendToRoutes is false ---------------

connections: # [array] Use QuickConnect - Direct connections to Destinations, and
  - pipeline: # [string] Pipeline or Pack
    output: # [string] Destination

# -------------------------------------------------------

environment: # [string] Environment - Optionally, enable this config only on a sp
pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

# -------------- if pqEnabled is true ---------------

pq: # [object]
  mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
  maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
  commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
  maxFileSize: # [string] File size limit - The maximum size to store in each que
  maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
  path: # [string] Queue file path - The location for the persistent queue files.
  compress: # [string] Compression - Codec to use to compress the persisted data

# -------------------------------------------------------

streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
```

```
exec_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled
  command: # [string] Command - Command to execute; supports Bourne shell (or CMD c
  retries: # [number] Retry limit - Maximum number of retry attempts in the event t
  scheduleType: # [string] Schedule type - Select a schedule type; either an interv

  # -------------- if scheduleType is interval ---------------

  interval: # [number; minimum: 1] Interval - Interval between command executions i

  # -----------------------------------------------------



  # -------------- if scheduleType is cronSchedule ---------------

  cronSchedule: # [string] Schedule - Cron schedule to execute the command on.

  # -----------------------------------------------------

  breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event-br
  staleChannelFlushMs: # [number; minimum: 10, maximum: 43200000] Event Breaker buf
  metadata: # [array] Fields - Fields to add to events from this input
    - name: # [string] Field Name
      value: # [string] Value - JavaScript expression to compute field's value, enc
  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Use QuickConnect - Direct connections to Destinations, and
    - pipeline: # [string] Pipeline or Pack
      output: # [string] Destination

  # -----------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a sp
  pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
```

```
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # ------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
firehose_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled
  host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses
  port: # [number; maximum: 65535] [required] Port - Port to listen on
  authTokens: # [array of strings] Auth tokens - Shared secrets to be provided by a
  tls: # [object] TLS settings (server side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    certificateName: # [string] Certificate - The name of the predefined certificat
    privKeyPath: # [string] Private key path - Path on server containing the privat
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
    certPath: # [string] Certificate path - Path on server containing certificates
    caPath: # [string] CA certificate path - Path on server containing CA certifica
    requestCert: # [boolean] Authenticate client (mutual auth) - Require clients to
    minVersion: # [string] Minimum TLS version
    maxVersion: # [string] Maximum TLS version

    # ------------------------------------------------------

  maxActiveReq: # [number] Active request limit - Maximum number of active requests
  maxRequestsPerSocket: # [integer] Requests-per-socket limit - Maximum number of r
  enableProxyHeader: # [boolean] Show originating IP - Extract the client IP and po
  captureHeaders: # [boolean] Capture request headers - Add request headers to ever
  activityLogSampleRate: # [number; minimum: 1] Activity log sample rate - How ofte
  requestTimeout: # [number] Request timeout (seconds) - How long to wait for an in
  socketTimeout: # [number] Socket timeout (seconds) - How long Cribl Edge should w
  keepAliveTimeout: # [number; minimum: 1, maximum: 600] Keep-alive timeout (second
  enableHealthCheck: # [boolean] Health check endpoint - Expose the /cribl_health e
  ipAllowlistRegex: # [string] IP allowlist regex - Messages from matched IP addres
```

```
  ipDenylistRegex: # [string] IP denylist regex - Messages from matched IP addresse
  metadata: # [array] Fields - Fields to add to events from this input
    - name: # [string] Field Name
      value: # [string] Value - JavaScript expression to compute field's value, enc
  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Use QuickConnect - Direct connections to Destinations, and
    - pipeline: # [string] Pipeline or Pack
      output: # [string] Destination

  # -------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a sp
  pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # -------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
google_pubsub_input: # [object]
  type: # [string] Input Type
  topicName: # [string] [required] Topic ID - ID of the topic to receive events fro
  subscriptionName: # [string] Subscription ID - ID of the subscription to use wher
  monitorSubscription: # [boolean] Monitor subscription for new messages - Use wher
  createTopic: # [boolean] Create topic - Create topic if it does not exist
  createSubscription: # [boolean] Create subscription - Create subscription if it o

  # -------------- if createSubscription is true ---------------
```

```
orderedDelivery: # [boolean] Ordered delivery - Receive events in the order they

# ------------------------------------------------------

region: # [string] Region - Region to retrieve messages from. Select 'default' to
googleAuthMethod: # [string] Google authentication method - Choose Auto to use Go

# -------------- if googleAuthMethod is manual ---------------

serviceAccountCredentials: # [string] Service account credentials - Contents of s

# ------------------------------------------------------


# -------------- if googleAuthMethod is secret ---------------

secret: # [string] Service account credentials (text secret) - Select or create a

# ------------------------------------------------------

maxBacklog: # [number; minimum: 1] Backlog limit - If Destination exerts backpres
concurrency: # [number; minimum: 1, maximum: 100] Number of concurrent streams -
requestTimeout: # [number; minimum: 10000] Request timeout (ms) - Pull request ti
metadata: # [array] Fields - Fields to add to events from this input
  - name: # [string] Field Name
    value: # [string] Value - JavaScript expression to compute field's value, enc
description: # [string] Description
disabled: # [boolean] Disabled
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

# -------------- if sendToRoutes is false ---------------

connections: # [array] Use QuickConnect - Direct connections to Destinations, and
  - pipeline: # [string] Pipeline or Pack
    output: # [string] Destination

# ------------------------------------------------------

environment: # [string] Environment - Optionally, enable this config only on a sp
pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

# -------------- if pqEnabled is true ---------------
```

```
  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # ------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
cribl_input: # [object]
  type: # [string] Input Type
  filter: # [string]
  metadata: # [array] Fields - Fields to add to events from this input
    - name: # [string] Field Name
      value: # [string] Value - JavaScript expression to compute field's value, enc
  description: # [string] Description
  disabled: # [boolean] Disabled
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Use QuickConnect - Direct connections to Destinations, and
    - pipeline: # [string] Pipeline or Pack
      output: # [string] Destination

  # ------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a sp
  pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
```

```
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # ------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
cribl_tcp_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled
  host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses
  port: # [number; maximum: 65535] [required] Port - Port to listen on
  tls: # [object] TLS settings (server side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    certificateName: # [string] Certificate - The name of the predefined certificat
    privKeyPath: # [string] Private key path - Path on server containing the privat
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
    certPath: # [string] Certificate path - Path on server containing certificates
    caPath: # [string] CA certificate path - Path on server containing CA certifica
    requestCert: # [boolean] Authenticate client (mutual auth) - Require clients to
    minVersion: # [string] Minimum TLS version
    maxVersion: # [string] Maximum TLS version

    # ------------------------------------------------------

  maxActiveCxn: # [number] Active connection limit - Maximum number of active conne
  socketIdleTimeout: # [number] Socket idle timeout (seconds) - How long Cribl Edge
  socketEndingMaxWait: # [number] Forced socket termination timeout (seconds) - How
  socketMaxLifespan: # [number] Socket max lifespan (seconds) - The maximum duratio
  enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection i
  metadata: # [array] Fields - Fields to add to events from this input
    - name: # [string] Field Name
      value: # [string] Value - JavaScript expression to compute field's value, enc
  enableLoadBalancing: # [boolean] Enable load balancing - Load balance traffic acr
  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Use QuickConnect - Direct connections to Destinations, and
```

```
    - pipeline: # [string] Pipeline or Pack
      output: # [string] Destination

  # ------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a sp
  pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # ------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
cribl_http_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled
  host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses
  port: # [number; maximum: 65535] [required] Port - Port to listen on
  authTokens: # [array of strings] Auth tokens - Shared secrets to be provided by a
  tls: # [object] TLS settings (server side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    certificateName: # [string] Certificate - The name of the predefined certificat
    privKeyPath: # [string] Private key path - Path on server containing the privat
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
    certPath: # [string] Certificate path - Path on server containing certificates
    caPath: # [string] CA certificate path - Path on server containing CA certifica
    requestCert: # [boolean] Authenticate client (mutual auth) - Require clients to
    minVersion: # [string] Minimum TLS version
    maxVersion: # [string] Maximum TLS version

    # ------------------------------------------------------
```

```
maxActiveReq: # [number] Active request limit - Maximum number of active requests
maxRequestsPerSocket: # [integer] Requests-per-socket limit - Maximum number of r
enableProxyHeader: # [boolean] Show originating IP - Extract the client IP and po
captureHeaders: # [boolean] Capture request headers - Add request headers to ever
activityLogSampleRate: # [number; minimum: 1] Activity log sample rate - How ofte
requestTimeout: # [number] Request timeout (seconds) - How long to wait for an ir
socketTimeout: # [number] Socket timeout (seconds) - How long Cribl Edge should w
keepAliveTimeout: # [number; minimum: 1, maximum: 600] Keep-alive timeout (second
enableHealthCheck: # [boolean] Health check endpoint - Expose the /cribl_health e
ipAllowlistRegex: # [string] IP allowlist regex - Messages from matched IP addres
ipDenylistRegex: # [string] IP denylist regex - Messages from matched IP addresse
metadata: # [array] Fields - Fields to add to events from this input
  - name: # [string] Field Name
    value: # [string] Value - JavaScript expression to compute field's value, enc
description: # [string] Description
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

# -------------- if sendToRoutes is false ---------------

connections: # [array] Use QuickConnect - Direct connections to Destinations, and
  - pipeline: # [string] Pipeline or Pack
    output: # [string] Destination

# -------------------------------------------------------

environment: # [string] Environment - Optionally, enable this config only on a sp
pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

# -------------- if pqEnabled is true ---------------

pq: # [object]
  mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
  maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
  commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
  maxFileSize: # [string] File size limit - The maximum size to store in each que
  maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
  path: # [string] Queue file path - The location for the persistent queue files.
  compress: # [string] Compression - Codec to use to compress the persisted data

# -------------------------------------------------------
```

```
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
cribl_lake_http_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled
  host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses
  port: # [number; maximum: 65535] [required] Port - Port to listen on
  authTokens: # [array of strings] Auth tokens - Shared secrets to be provided by a
  tls: # [object] TLS settings (server side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    certificateName: # [string] Certificate - The name of the predefined certificat
    privKeyPath: # [string] Private key path - Path on server containing the privat
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
    certPath: # [string] Certificate path - Path on server containing certificates
    caPath: # [string] CA certificate path - Path on server containing CA certifica
    requestCert: # [boolean] Authenticate client (mutual auth) - Require clients to
    minVersion: # [string] Minimum TLS version
    maxVersion: # [string] Maximum TLS version

    # ---------------------------------------------------------

  maxActiveReq: # [number] Active request limit - Maximum number of active requests
  maxRequestsPerSocket: # [integer] Requests-per-socket limit - Maximum number of r
  enableProxyHeader: # [boolean] Show originating IP - Extract the client IP and po
  captureHeaders: # [boolean] Capture request headers - Add request headers to ever
  activityLogSampleRate: # [number; minimum: 1] Activity log sample rate - How ofte
  requestTimeout: # [number] Request timeout (seconds) - How long to wait for an ir
  socketTimeout: # [number] Socket timeout (seconds) - How long Cribl Edge should w
  keepAliveTimeout: # [number; minimum: 1, maximum: 600] Keep-alive timeout (second
  enableHealthCheck: # [boolean] Health check endpoint - Expose the /cribl_health e
  ipAllowlistRegex: # [string] IP allowlist regex - Messages from matched IP addres
  ipDenylistRegex: # [string] IP denylist regex - Messages from matched IP addresse
  criblAPI: # [string] Cribl HTTP event API - Absolute path on which to listen for
  elasticAPI: # [string] Elasticsearch API endpoint (Bulk API) - Absolute path on w
  splunkHecAPI: # [string] Splunk HEC endpoint - Absolute path on which listen for
  splunkHecAcks: # [boolean] Enable Splunk HEC acknowledgements
  metadata: # [array] Fields - Fields to add to events from this input
    - name: # [string] Field Name
      value: # [string] Value - JavaScript expression to compute field's value, enc
  authTokensExt: # [array] Auth tokens
    - token: # [string] Token
```

```
    description: # [string]
    metadata: # [array] Fields - Fields to add to events referencing this token
    - name: # [string] Field Name
      value: # [string] Value - JavaScript expression to compute field's value, e
    splunkHecMetadata: # [object]
    enabled: # [boolean]

    # -------------- if enabled is true ---------------

    defaultDataset: # [string]
    allowedIndexesAtToken: # [array of strings]

    # --------------------------------------------------------

    elasticsearchMetadata: # [object]
    enabled: # [boolean]

    # -------------- if enabled is true ---------------

    defaultDataset: # [string]

    # --------------------------------------------------------

description: # [string] Description
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

# -------------- if sendToRoutes is false ---------------

connections: # [array] Use QuickConnect - Direct connections to Destinations, and
  - pipeline: # [string] Pipeline or Pack
    output: # [string] Destination

# --------------------------------------------------------

environment: # [string] Environment - Optionally, enable this config only on a sp
pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

# -------------- if pqEnabled is true ---------------

pq: # [object]
  mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
  maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
```

```
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # ------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
tcpjson_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled
  host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses
  port: # [number; maximum: 65535] [required] Port - Port to listen on
  tls: # [object] TLS settings (server side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    certificateName: # [string] Certificate - The name of the predefined certificat
    privKeyPath: # [string] Private key path - Path on server containing the privat
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
    certPath: # [string] Certificate path - Path on server containing certificates
    caPath: # [string] CA certificate path - Path on server containing CA certifica
    requestCert: # [boolean] Authenticate client (mutual auth) - Require clients to
    minVersion: # [string] Minimum TLS version
    maxVersion: # [string] Maximum TLS version

    # ------------------------------------------------------

  ipWhitelistRegex: # [string] IP allowlist regex - Regex matching IP addresses tha
  maxActiveCxn: # [number] Active connection limit - Maximum number of active conne
  socketIdleTimeout: # [number] Socket idle timeout (seconds) - How long Cribl Edge
  socketEndingMaxWait: # [number] Forced socket termination timeout (seconds) - How
  socketMaxLifespan: # [number] Socket max lifespan (seconds) - The maximum duratio
  enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection i
  metadata: # [array] Fields - Fields to add to events from this input
    - name: # [string] Field Name
      value: # [string] Value - JavaScript expression to compute field's value, enc
  enableLoadBalancing: # [boolean] Enable load balancing - Load balance traffic acr
  authType: # [string] Authentication method - Select Manual to enter an auth token
  authToken: # [string] Auth token - Shared secret to be provided by any client (ir
```

```
  # -------------- if authType is manual ---------------


  # ------------------------------------------------------

  textSecret: # [string] Auth token (text secret) - Select or create a stored text

  # -------------- if authType is secret ---------------


  # ------------------------------------------------------

  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Use QuickConnect - Direct connections to Destinations, and
    - pipeline: # [string] Pipeline or Pack
      output: # [string] Destination

  # ------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a sp
  pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # ------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
system_metrics_input: # [object]
  type: # [string] Input Type
```

```
interval: # [number; minimum: 1] Polling interval - Time, in seconds, between cor
host: # [object]
  mode: # [string]  - Select level of detail for host metrics

  # -------------- if mode is custom ---------------

  custom: # [object]
    system: # [object]
      mode: # [string]  - Select the level of detail for system metrics

      # -------------- if mode is custom ---------------

      processes: # [boolean] Process metrics - Generate metrics for the numbers c

      # ------------------------------------------------------

    cpu: # [object]
      mode: # [string]  - Select the level of detail for CPU metrics

      # -------------- if mode is custom ---------------

      perCpu: # [boolean] Per-CPU metrics - Generate metrics for each CPU
      detail: # [boolean] Detailed metrics - Generate metrics for all CPU states
      time: # [boolean] CPU time metrics - Generate raw, monotonic CPU time count

      # ------------------------------------------------------

    memory: # [object]
      mode: # [string]  - Select the level of detail for memory metrics

      # -------------- if mode is custom ---------------

      detail: # [boolean] Detailed metrics - Generate metrics for all memory stat

      # ------------------------------------------------------

    network: # [object]
      mode: # [string]  - Select the level of detail for network metrics

      # -------------- if mode is custom ---------------

      devices: # [array of strings] Interface filter - Network interfaces to incl
      perInterface: # [boolean] Per-interface metrics - Generate separate metrics
```

```
      detail: # [boolean] Detailed metrics - Generate full network metrics

      # ------------------------------------------------------

    disk: # [object]
      mode: # [string]  - Select the level of detail for disk metrics

      # -------------- if mode is custom ---------------

      devices: # [array of strings] Device filter - Block devices to include/excl
      mountpoints: # [array of strings] Mountpoint filter - Filesystem mountpoint
      fstypes: # [array of strings] Filesystem type filter - Filesystem types to
      perDevice: # [boolean] Per-device metrics - Generate separate metrics for e
      detail: # [boolean] Detailed metrics - Generate full disk metrics

      # ------------------------------------------------------


  # --------------------------------------------------------

process: # [object]
  sets: # [array] Process sets - Configure sets to collect process metrics
    - name: # [string] Set Name
      filter: # [string] Filter Expression
      includeChildren: # [boolean] Include Child Processes
container: # [object]
  mode: # [string]  - Select the level of detail for container metrics

  # -------------- if mode is basic ---------------

  dockerSocket: # [array of strings] Docker socket - Full paths for Docker's UNI
  dockerTimeout: # [number; minimum: 1] Docker timeout - Timeout, in seconds, for

  # ------------------------------------------------------


  # -------------- if mode is all ---------------

  dockerSocket: # [array of strings] Docker socket - Full paths for Docker's UNI
  dockerTimeout: # [number; minimum: 1] Docker timeout - Timeout, in seconds, for

  # ------------------------------------------------------
```

```
  # -------------- if mode is custom ---------------

  filters: # [array] Container filters - Containers matching any of these will be
    - expr: # [string] Expression
  allContainers: # [boolean] All containers - Include stopped and paused containe
  perDevice: # [boolean] Per-device metrics - Generate separate metrics for each
  detail: # [boolean] Detailed metrics - Generate full container metrics
  dockerSocket: # [array of strings] Docker socket - Full paths for Docker's UNI
  dockerTimeout: # [number; minimum: 1] Docker timeout - Timeout, in seconds, for

  # ------------------------------------------------------

metadata: # [array] Fields - Fields to add to events from this input
  - name: # [string] Field Name
    value: # [string] Value - JavaScript expression to compute field's value, enc
persistence: # [object] persistence
  enable: # [boolean] Enable disk spooling - Spool metrics to disk for Cribl Edge

  # -------------- if enable is true ---------------

  timeWindow: # [string] Bucket time span - Time span for each file bucket
  maxDataSize: # [string] Data size limit - Maximum disk space allowed to be cons
  maxDataTime: # [string] Data age limit - Maximum amount of time to retain data
  compress: # [string] Data compression format
  destPath: # [string] Path location - Path to use to write metrics. Defaults to

  # ------------------------------------------------------

description: # [string] Description
disabled: # [boolean] Disabled
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

# -------------- if sendToRoutes is false ---------------

connections: # [array] Use QuickConnect - Direct connections to Destinations, and
  - pipeline: # [string] Pipeline or Pack
    output: # [string] Destination

# -------------------------------------------------------

environment: # [string] Environment - Optionally, enable this config only on a sp
```

```
  pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # ---------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
system_state_input: # [object]
  type: # [string] Input Type
  interval: # [number; minimum: 1] Polling interval - Time, in seconds, between cor
  metadata: # [array] Fields - Fields to add to events from this input
    - name: # [string] Field Name
      value: # [string] Value - JavaScript expression to compute field's value, enc
  collectors: # [object]
    hostsfile: # [object] Hosts File - Creates events based on entries collected fr
      enable: # [boolean] Enabled
    interfaces: # [object] Interfaces - Creates events for each of the host's netwo
      enable: # [boolean] Enabled
    disk: # [object] Disks & File Systems - Creates events for physical disks, part
      enable: # [boolean] Enabled
    metadata: # [object] Host Info - Creates events based on the host system's curr
      enable: # [boolean] Enabled
    routes: # [object] Routes - Creates events based on entries collected from the
      enable: # [boolean] Enabled
    dns: # [object] DNS - Creates events for DNS resolvers and search entries
      enable: # [boolean] Enabled
    user: # [object] Users & Groups - Creates events for local users and groups
      enable: # [boolean] Enabled
    firewall: # [object] Firewall - Creates events for Firewall rules entries
      enable: # [boolean] Enabled
    services: # [object] Services - Creates events from the list of services
      enable: # [boolean] Enabled
    ports: # [object] Listening Ports - Creates events from list of listening ports
      enable: # [boolean] Enabled
```

```
  loginUsers: # [object] Logged-In Users - Creates events from list of logged-in
    enable: # [boolean] Enabled
persistence: # [object]
  enable: # [boolean] Enable disk spooling - Spool metrics to disk for Cribl Edge

  # -------------- if enable is true ---------------

  timeWindow: # [string] Bucket time span - Time span for each file bucket
  maxDataSize: # [string] Data size limit - Maximum disk space allowed to be cons
  maxDataTime: # [string] Data age limit - Maximum amount of time to retain data
  compress: # [string] Data compression format
  destPath: # [string] Path location - Path to use to write metrics. Defaults to

  # ------------------------------------------------------

disableNativeModule: # [boolean] Use Windows Tools - Enable to use built-in tools
description: # [string] Description
disabled: # [boolean] Disabled
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

# -------------- if sendToRoutes is false ---------------

connections: # [array] Use QuickConnect - Direct connections to Destinations, and
  - pipeline: # [string] Pipeline or Pack
    output: # [string] Destination

# -------------------------------------------------------

environment: # [string] Environment - Optionally, enable this config only on a sp
pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

# -------------- if pqEnabled is true ---------------

pq: # [object]
  mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
  maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
  commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
  maxFileSize: # [string] File size limit - The maximum size to store in each que
  maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
  path: # [string] Queue file path - The location for the persistent queue files.
  compress: # [string] Compression - Codec to use to compress the persisted data
```

```
  # ------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
kube_metrics_input: # [object]
  type: # [string] Input Type
  interval: # [number; minimum: 1] Polling interval - Time, in seconds, between cor
  rules: # [array] Filter Rules - Add rules to decide which Kubernetes objects to g
    - filter: # [string] Filter Expression - JavaScript expression applied to Kuber
      description: # [string] Description - Optional description of this rule's pur
  metadata: # [array] Fields - Fields to add to events from this input
    - name: # [string] Field Name
      value: # [string] Value - JavaScript expression to compute field's value, enc
  persistence: # [object] persistence
    enable: # [boolean] Enable disk spooling - Spool metrics on disk for Cribl Sear

    # -------------- if enable is true ---------------

    timeWindow: # [string] Bucket time span - Time span for each file bucket
    maxDataSize: # [string] Data size limit - Maximum disk space allowed to be cons
    maxDataTime: # [string] Data age limit - Maximum amount of time to retain data
    compress: # [string] Data compression format
    destPath: # [string] Path location - Path to use to write metrics. Defaults to

    # ------------------------------------------------------

  description: # [string] Description
  disabled: # [boolean] Disabled
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Use QuickConnect - Direct connections to Destinations, and
    - pipeline: # [string] Pipeline or Pack
      output: # [string] Destination

  # ------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a sp
  pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

  # -------------- if pqEnabled is true ---------------
```

```
  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # ------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
kube_logs_input: # [object]
  type: # [string] Input Type
  interval: # [number; minimum: 1] Polling interval - Time, in seconds, between che
  rules: # [array] Filter Rules - Add rules to decide which Pods to collect logs fr
    - filter: # [string] Filter Expression - JavaScript expression applied to Pod o
      description: # [string] Description - Optional description of this rule's pur
  timestamps: # [boolean] Enable timestamps - For use when containers do not emit a
  metadata: # [array] Fields - Fields to add to events from this input
    - name: # [string] Field Name
      value: # [string] Value - JavaScript expression to compute field's value, enc
  persistence: # [object] Disk Spooling
    enable: # [boolean] Enable disk spooling - Spool events on disk for Cribl Edge

    # -------------- if enable is true ---------------

    timeWindow: # [string] Bucket time span - Time period for grouping spooled even
    maxDataSize: # [string] Data size limit - Maximum disk space that can be consum
    maxDataTime: # [string] Data age limit - Maximum amount of time to retain data
    compress: # [string] Compression - Data compression format. Default is gzip.

    # ------------------------------------------------------

  breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event-br
  staleChannelFlushMs: # [number; minimum: 10, maximum: 43200000] Event Breaker buf
  enableLoadBalancing: # [boolean] Enable load balancing - Load balance traffic acr
  description: # [string] Description
  disabled: # [boolean] Disabled
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

  # -------------- if sendToRoutes is false ---------------
```

```
  connections: # [array] Use QuickConnect - Direct connections to Destinations, and
    - pipeline: # [string] Pipeline or Pack
      output: # [string] Destination

  # ------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a sp
  pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # ------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
kube_events_input: # [object]
  type: # [string] Input Type
  rules: # [array] Filter Rules - Filtering on event fields
    - filter: # [string] Filter Expression - JavaScript expression applied to Kuber
      description: # [string] Description - Optional description of this rule's pur
  metadata: # [array] Fields - Fields to add to events from this input
    - name: # [string] Field Name
      value: # [string] Value - JavaScript expression to compute field's value, enc
  description: # [string] Description
  disabled: # [boolean] Disabled
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Use QuickConnect - Direct connections to Destinations, and
    - pipeline: # [string] Pipeline or Pack
      output: # [string] Destination
```

```
  # ------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a sp
  pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # ------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
windows_metrics_input: # [object]
  type: # [string] Input Type
  interval: # [number; minimum: 1] Polling interval - Time, in seconds, between cor
  host: # [object]
    mode: # [string]  - Select level of detail for host metrics

    # -------------- if mode is custom ---------------

    custom: # [object]
      system: # [object]
        mode: # [string]  - Select the level of details for system metrics

        # -------------- if mode is custom ---------------

        detail: # [boolean] Detailed metrics - Generate metrics for all system info

        # ------------------------------------------------------

      cpu: # [object]
        mode: # [string]  - Select the level of details for CPU metrics

        # -------------- if mode is custom ---------------

        perCpu: # [boolean] Per-CPU metrics - Generate metrics for each CPU
```

```
      detail: # [boolean] Detailed metrics - Generate metrics for all CPU states
      time: # [boolean] CPU time metrics - Generate raw, monotonic CPU time count

      # ------------------------------------------------------

    memory: # [object]
      mode: # [string]  - Select the level of details for memory metrics

      # -------------- if mode is custom ---------------

      detail: # [boolean] Detailed metrics - Generate metrics for all memory stat

      # ------------------------------------------------------

    network: # [object]
      mode: # [string]  - Select the level of details for network metrics

      # -------------- if mode is custom ---------------

      devices: # [array of strings] Interface filter - Network interfaces to incl
      perInterface: # [boolean] Per interface metrics - Generate separate metrics
      detail: # [boolean] Detailed metrics - Generate full network metrics

      # ------------------------------------------------------

    disk: # [object]
      mode: # [string]  - Select the level of details for disk metrics

      # -------------- if mode is custom ---------------

      volumes: # [array of strings] Volume filter - Windows volumes to include/ex
      perVolume: # [boolean] Per volume metrics - Generate separate metrics for e

      # ------------------------------------------------------


  # ------------------------------------------------------

process: # [object]
  sets: # [array] Process sets - Configure sets to collect process metrics
    - name: # [string] Set Name
      filter: # [string] Filter Expression
      includeChildren: # [boolean] Include Child Processes
```

```
metadata: # [array] Fields - Fields to add to events from this input
  - name: # [string] Field Name
    value: # [string] Value - JavaScript expression to compute field's value, enc
persistence: # [object] persistence
  enable: # [boolean] Enable disk spooling - Spool metrics to disk for Cribl Edge

  # -------------- if enable is true ---------------

  timeWindow: # [string] Bucket time span - Time span for each file bucket
  maxDataSize: # [string] Data size limit - Maximum disk space allowed to be cons
  maxDataTime: # [string] Data age limit - Maximum amount of time to retain data
  compress: # [string] Data compression format
  destPath: # [string] Path location - Path to use to write metrics. Defaults to

  # -----------------------------------------------------

disableNativeModule: # [boolean] Use Windows Tools - Enable to use built-in tools
description: # [string] Description
disabled: # [boolean] Disabled
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

# -------------- if sendToRoutes is false ---------------

connections: # [array] Use QuickConnect - Direct connections to Destinations, and
  - pipeline: # [string] Pipeline or Pack
    output: # [string] Destination

# -----------------------------------------------------

environment: # [string] Environment - Optionally, enable this config only on a sp
pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

# -------------- if pqEnabled is true ---------------

pq: # [object]
  mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
  maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
  commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
  maxFileSize: # [string] File size limit - The maximum size to store in each que
  maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
  path: # [string] Queue file path - The location for the persistent queue files.
  compress: # [string] Compression - Codec to use to compress the persisted data
```

```
  # ------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
crowdstrike_input: # [object]
  type: # [string] Input Type
  queueName: # [string] Queue - The name, URL, or ARN of the SQS queue to read noti
  fileFilter: # [string] Filename filter - Regex matching file names to download ar
  awsAccountId: # [string] AWS account ID - SQS queue owner's AWS account ID. Leave
  awsAuthenticationMethod: # [string] Authentication method - AWS authentication me

  # -------------- if awsAuthenticationMethod is manual ---------------

  awsApiKey: # [string] Access key

  # ------------------------------------------------------


  # -------------- if awsAuthenticationMethod is secret ---------------

  awsSecret: # [string] Secret key pair - Select or create a stored secret that ref

  # ------------------------------------------------------

  awsSecretKey: # [string] Secret key
  region: # [string] Region - AWS Region where the S3 bucket and SQS queue are loca
  endpoint: # [string] Endpoint - S3 service endpoint. If empty, defaults to the AW
  signatureVersion: # [string] Signature version - Signature version to use for sig
  reuseConnections: # [boolean] Reuse connections - Reuse connections between reque
  rejectUnauthorized: # [boolean] Reject unauthorized certificates - Reject certifi
  breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event-br
  staleChannelFlushMs: # [number; minimum: 10, maximum: 43200000] Event Breaker buf
  maxMessages: # [number; minimum: 1, maximum: 10] Message limit - The maximum numb
  visibilityTimeout: # [number; maximum: 43200] Visibility timeout seconds - After
  numReceivers: # [number; minimum: 1, maximum: 100] Number of receivers - How many
  socketTimeout: # [number; minimum: 1, maximum: 43200] Socket timeout - Socket ina
  skipOnError: # [boolean] Skip file on error - Skip files that trigger a processin
  includeSqsMetadata: # [boolean] Include notification metadata - Attach SQS notifi
  enableAssumeRole: # [boolean] Enable for Amazon S3 - Use Assume Role credentials
  assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the role
  assumeRoleExternalId: # [string] External ID - External ID to use when assuming r
  durationSeconds: # [number; minimum: 900, maximum: 43200] Duration (seconds) - Du
  enableSQSAssumeRole: # [boolean] Enable for Amazon SQS - Use Assume Role credenti
```

```
preprocess: # [object]
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  command: # [string] Command - Command to feed the data through (via stdin) and
  args: # [array of strings] Arguments - Arguments to be added to the custom comm

  # ---------------------------------------------------------

metadata: # [array] Fields - Fields to add to events from this input
  - name: # [string] Field Name
    value: # [string] Value - JavaScript expression to compute field's value, enc
checkpointing: # [object]
  enabled: # [boolean] Enable checkpointing - Resume processing files after an ir

  # -------------- if enabled is true ---------------

  retries: # [number; maximum: 100] Retries - The number of times to retry proces

  # ---------------------------------------------------------

pollTimeout: # [number; minimum: 1, maximum: 20] Poll timeout (secs) - How long t
encoding: # [string] Encoding - Character encoding to use when parsing ingested d
description: # [string] Description
disabled: # [boolean] Disabled
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

# -------------- if sendToRoutes is false ---------------

connections: # [array] Use QuickConnect - Direct connections to Destinations, and
  - pipeline: # [string] Pipeline or Pack
    output: # [string] Destination

# ---------------------------------------------------------

environment: # [string] Environment - Optionally, enable this config only on a sp
pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

# -------------- if pqEnabled is true ---------------

pq: # [object]
```

```
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # ------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
datadog_agent_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled
  host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses
  port: # [number; maximum: 65535] [required] Port - Port to listen on
  tls: # [object] TLS settings (server side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    certificateName: # [string] Certificate - The name of the predefined certificat
    privKeyPath: # [string] Private key path - Path on server containing the privat
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
    certPath: # [string] Certificate path - Path on server containing certificates
    caPath: # [string] CA certificate path - Path on server containing CA certifica
    requestCert: # [boolean] Authenticate client (mutual auth) - Require clients to
    minVersion: # [string] Minimum TLS version
    maxVersion: # [string] Maximum TLS version

    # ------------------------------------------------------

  maxActiveReq: # [number] Active request limit - Maximum number of active requests
  maxRequestsPerSocket: # [integer] Requests-per-socket limit - Maximum number of r
  enableProxyHeader: # [boolean] Show originating IP - Extract the client IP and po
  captureHeaders: # [boolean] Capture request headers - Add request headers to even
  activityLogSampleRate: # [number; minimum: 1] Activity log sample rate - How ofte
  requestTimeout: # [number] Request timeout (seconds) - How long to wait for an in
  socketTimeout: # [number] Socket timeout (seconds) - How long Cribl Edge should w
  keepAliveTimeout: # [number; minimum: 1, maximum: 600] Keep-alive timeout (second
  enableHealthCheck: # [boolean] Health check endpoint - Expose the /cribl_health e
  ipAllowlistRegex: # [string] IP allowlist regex - Messages from matched IP addres
  ipDenylistRegex: # [string] IP denylist regex - Messages from matched IP addresse
```

```
  extractMetrics: # [boolean] Extract metrics - Toggle to Yes to extract each incom
  metadata: # [array] Fields - Fields to add to events from this input
    - name: # [string] Field Name
      value: # [string] Value - JavaScript expression to compute field's value, enc
  proxyMode: # [object]
    enabled: # [boolean] Forward API key validation requests - Toggle to Yes to ser

    # -------------- if enabled is true ---------------

    rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to r

    # -------------------------------------------------------

  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Use QuickConnect - Direct connections to Destinations, and
    - pipeline: # [string] Pipeline or Pack
      output: # [string] Destination

  # -------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a sp
  pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # -------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
datagen_input: # [object]
```

```
  type: # [string] Input Type
  samples: # [array] Datagens
    - sample: # [string] Data Generator File Name
      eventsPerSec: # [number; minimum: 1] Events Per Second Per Worker Node - Maxi
  metadata: # [array] Fields - Fields to add to events from this input
    - name: # [string] Field Name
      value: # [string] Value - JavaScript expression to compute field's value, enc
  description: # [string] Description
  disabled: # [boolean] Disabled
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Use QuickConnect - Direct connections to Destinations, and
    - pipeline: # [string] Pipeline or Pack
      output: # [string] Destination

  # --------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a sp
  pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # --------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
http_raw_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled
  host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses
  port: # [number; maximum: 65535] [required] Port - Port to listen on
  authTokens: # [array of strings] Auth tokens - Shared secrets to be provided by a
```

```
tls: # [object] TLS settings (server side)
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  certificateName: # [string] Certificate - The name of the predefined certificat
  privKeyPath: # [string] Private key path - Path on server containing the privat
  passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
  certPath: # [string] Certificate path - Path on server containing certificates
  caPath: # [string] CA certificate path - Path on server containing CA certifica
  requestCert: # [boolean] Authenticate client (mutual auth) - Require clients to
  minVersion: # [string] Minimum TLS version
  maxVersion: # [string] Maximum TLS version

  # -------------------------------------------------------

maxActiveReq: # [number] Active request limit - Maximum number of active requests
maxRequestsPerSocket: # [integer] Requests-per-socket limit - Maximum number of r
enableProxyHeader: # [boolean] Show originating IP - Extract the client IP and po
captureHeaders: # [boolean] Capture request headers - Add request headers to ever
activityLogSampleRate: # [number; minimum: 1] Activity log sample rate - How ofte
requestTimeout: # [number] Request timeout (seconds) - How long to wait for an in
socketTimeout: # [number] Socket timeout (seconds) - How long Cribl Edge should w
keepAliveTimeout: # [number; minimum: 1, maximum: 600] Keep-alive timeout (second
enableHealthCheck: # [boolean] Health check endpoint - Expose the /cribl_health e
ipAllowlistRegex: # [string] IP allowlist regex - Messages from matched IP addres
ipDenylistRegex: # [string] IP denylist regex - Messages from matched IP addresse
breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event-br
staleChannelFlushMs: # [number; minimum: 10, maximum: 43200000] Event Breaker buf
metadata: # [array] Fields - Fields to add to events from this input
  - name: # [string] Field Name
    value: # [string] Value - JavaScript expression to compute field's value, enc
allowedPaths: # [array of strings] Allowed URI paths - List of URI paths accepted
allowedMethods: # [array of strings] Allowed HTTP methods - List of HTTP methods
authTokensExt: # [array] Auth tokens - Shared secrets to be provided by any clien
  - token: # [string] Token - Shared secret to be provided by any client (Authori
    description: # [string] Description
    metadata: # [array] Fields - Fields to add to events referencing this token
    - name: # [string] Field Name
      value: # [string] Value - JavaScript expression to compute field's value, e
description: # [string] Description
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t
```

```
  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Use QuickConnect - Direct connections to Destinations, and
    - pipeline: # [string] Pipeline or Pack
      output: # [string] Destination

  # --------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a sp
  pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # --------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
kinesis_input: # [object]
  type: # [string] Input Type
  streamName: # [string] Stream name - Kinesis Data Stream to read data from
  serviceInterval: # [number; minimum: 1, maximum: 5] Service period - Time interva
  shardExpr: # [string] Shard selection expression - A JavaScript expression to be
  shardIteratorType: # [string] Shard iterator start - Location at which to start r
  payloadFormat: # [string] Record data format - Format of data inside the Kinesis
  getRecordsLimit: # [number; minimum: 5000, maximum: 10000] Records limit per call
  getRecordsLimitTotal: # [number; minimum: 20000] Total records limit - Maximum nu
  loadBalancingAlgorithm: # [string] Shard load balancing - The load-balancing algo
  awsAuthenticationMethod: # [string] Authentication method - AWS authentication me

  # -------------- if awsAuthenticationMethod is manual ---------------

  awsApiKey: # [string] Access key

  # --------------------------------------------------------
```

```
# -------------- if awsAuthenticationMethod is secret ---------------

awsSecret: # [string] Secret key pair - Select or create a stored secret that ref

# -------------------------------------------------------

awsSecretKey: # [string] Secret key
region: # [string] [required] Region - Region where the Kinesis stream is located
endpoint: # [string] Endpoint - Kinesis stream service endpoint. If empty, defaul
signatureVersion: # [string] Signature version - Signature version to use for sig
reuseConnections: # [boolean] Reuse connections - Reuse connections between reque
rejectUnauthorized: # [boolean] Reject unauthorized certificates - Reject certifi
enableAssumeRole: # [boolean] Enable for Kinesis stream - Use Assume Role credent
assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the role
assumeRoleExternalId: # [string] External ID - External ID to use when assuming r
durationSeconds: # [number; minimum: 900, maximum: 43200] Duration (seconds) - Du
verifyKPLCheckSums: # [boolean] Verify KPL checksums - Verify Kinesis Producer Li
avoidDuplicates: # [boolean] Avoid duplicate records - When resuming streaming fr
metadata: # [array] Fields - Fields to add to events from this input
  - name: # [string] Field Name
    value: # [string] Value - JavaScript expression to compute field's value, enc
description: # [string] Description
disabled: # [boolean] Disabled
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

# -------------- if sendToRoutes is false ---------------

connections: # [array] Use QuickConnect - Direct connections to Destinations, and
  - pipeline: # [string] Pipeline or Pack
    output: # [string] Destination

# -------------------------------------------------------

environment: # [string] Environment - Optionally, enable this config only on a sp
pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

# -------------- if pqEnabled is true ---------------

pq: # [object]
  mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
```

```
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # ------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
criblmetrics_input: # [object]
  type: # [string] Input Type
  prefix: # [string] Metric name prefix - A prefix that is applied to the metrics p
  fullFidelity: # [boolean] Full fidelity - Include granular metrics. Disabling thi
  metadata: # [array] Fields - Fields to add to events from this input
    - name: # [string] Field Name
      value: # [string] Value - JavaScript expression to compute field's value, enc
  description: # [string] Description
  disabled: # [boolean] Disabled
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Use QuickConnect - Direct connections to Destinations, and
    - pipeline: # [string] Pipeline or Pack
      output: # [string] Destination

  # -------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a sp
  pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data
```

```
  # ------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
metrics_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled
  host: # [string] Address - Address to bind on. For IPv4 (all addresses), use the
  udpPort: # [number; maximum: 65535] UDP Port - Enter UDP port number to listen or
  tcpPort: # [number; maximum: 65535] TCP Port - Enter TCP port number to listen or
  maxBufferSize: # [number] Buffer size limit (events) - Maximum number of events t
  ipWhitelistRegex: # [string] IP allowlist regex - Regex matching IP addresses tha
  enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection i
  tls: # [object] TLS settings (server side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    certificateName: # [string] Certificate - The name of the predefined certificat
    privKeyPath: # [string] Private key path - Path on server containing the privat
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
    certPath: # [string] Certificate path - Path on server containing certificates
    caPath: # [string] CA certificate path - Path on server containing CA certifica
    requestCert: # [boolean] Authenticate client (mutual auth) - Require clients to
    minVersion: # [string] Minimum TLS version
    maxVersion: # [string] Maximum TLS version

    # ------------------------------------------------------

  metadata: # [array] Fields - Fields to add to events from this input
    - name: # [string] Field Name
      value: # [string] Value - JavaScript expression to compute field's value, enc
  udpSocketRxBufSize: # [number; minimum: 256, maximum: 4294967295] UDP socket buff
  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Use QuickConnect - Direct connections to Destinations, and
    - pipeline: # [string] Pipeline or Pack
      output: # [string] Destination
```

```
  # ------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a sp
  pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # ------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
s3_input: # [object]
  type: # [string] Input Type
  queueName: # [string] Queue - The name, URL, or ARN of the SQS queue to read noti
  fileFilter: # [string] Filename filter - Regex matching file names to download ar
  awsAccountId: # [string] AWS account ID - SQS queue owner's AWS account ID. Leave
  awsAuthenticationMethod: # [string] Authentication method - AWS authentication me

  # -------------- if awsAuthenticationMethod is manual ---------------

  awsApiKey: # [string] Access key

  # ------------------------------------------------------



  # -------------- if awsAuthenticationMethod is secret ---------------

  awsSecret: # [string] Secret key pair - Select or create a stored secret that ref

  # ------------------------------------------------------

  awsSecretKey: # [string] Secret key
  region: # [string] Region - AWS Region where the S3 bucket and SQS queue are loca
  endpoint: # [string] Endpoint - S3 service endpoint. If empty, defaults to the AW
  signatureVersion: # [string] Signature version - Signature version to use for sig
```

```
reuseConnections: # [boolean] Reuse connections - Reuse connections between reque
rejectUnauthorized: # [boolean] Reject unauthorized certificates - Reject certifi
breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event-br
staleChannelFlushMs: # [number; minimum: 10, maximum: 43200000] Event Breaker buf
maxMessages: # [number; minimum: 1, maximum: 10] Message limit - The maximum numb
visibilityTimeout: # [number; maximum: 43200] Visibility timeout seconds - After
numReceivers: # [number; minimum: 1, maximum: 100] Number of receivers - How many
socketTimeout: # [number; minimum: 1, maximum: 43200] Socket timeout - Socket ina
skipOnError: # [boolean] Skip file on error - Skip files that trigger a processir
includeSqsMetadata: # [boolean] Include notification metadata - Attach SQS notifi
enableAssumeRole: # [boolean] Enable for Amazon S3 - Use Assume Role credentials
assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the role
assumeRoleExternalId: # [string] External ID - External ID to use when assuming r
durationSeconds: # [number; minimum: 900, maximum: 43200] Duration (seconds) - Du
enableSQSAssumeRole: # [boolean] Enable for Amazon SQS - Use Assume Role credenti
preprocess: # [object]
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  command: # [string] Command - Command to feed the data through (via stdin) and
  args: # [array of strings] Arguments - Arguments to be added to the custom comm

  # ---------------------------------------------------

metadata: # [array] Fields - Fields to add to events from this input
  - name: # [string] Field Name
    value: # [string] Value - JavaScript expression to compute field's value, enc
parquetChunkSizeMB: # [number; minimum: 1, maximum: 100] Parquet chunk size limit
parquetChunkDownloadTimeout: # [number; minimum: 1, maximum: 3600] Parquet chunk
checkpointing: # [object]
  enabled: # [boolean] Enable checkpointing - Resume processing files after an ir

  # -------------- if enabled is true ---------------

  retries: # [number; maximum: 100] Retries - The number of times to retry proces

  # ---------------------------------------------------

pollTimeout: # [number; minimum: 1, maximum: 20] Poll timeout (secs) - How long t
encoding: # [string] Encoding - Character encoding to use when parsing ingested d
tagAfterProcessing: # [boolean] Tag after processing - Add a tag to processed S3
```

```
  # -------------- if tagAfterProcessing is true ---------------

  processedTagKey: # [string] Tag key - The key for the S3 object tag applied after
  processedTagValue: # [string] Tag value - The value for the S3 object tag applied

  # ------------------------------------------------------

  description: # [string] Description
  disabled: # [boolean] Disabled
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Use QuickConnect - Direct connections to Destinations, and
    - pipeline: # [string] Pipeline or Pack
      output: # [string] Destination

  # ------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a sp
  pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # ------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
s3_inventory_input: # [object]
  queueName: # [string] Queue - The name, URL, or ARN of the SQS queue to read noti
  fileFilter: # [string] Filename filter - Regex matching file names to download ar
  awsAccountId: # [string] AWS account ID - SQS queue owner's AWS account ID. Leave
  awsAuthenticationMethod: # [string] Authentication method - AWS authentication me
```

```
# -------------- if awsAuthenticationMethod is manual ---------------

awsApiKey: # [string] Access key

# -------------------------------------------------------


# -------------- if awsAuthenticationMethod is secret ---------------

awsSecret: # [string] Secret key pair - Select or create a stored secret that ref

# -------------------------------------------------------


awsSecretKey: # [string] Secret key
region: # [string] Region - AWS Region where the S3 bucket and SQS queue are loca
endpoint: # [string] Endpoint - S3 service endpoint. If empty, defaults to the AW
signatureVersion: # [string] Signature version - Signature version to use for sig
reuseConnections: # [boolean] Reuse connections - Reuse connections between reque
rejectUnauthorized: # [boolean] Reject unauthorized certificates - Reject certifi
breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event-br
staleChannelFlushMs: # [number; minimum: 10, maximum: 43200000] Event Breaker buf
maxMessages: # [number; minimum: 1, maximum: 10] Message limit - The maximum numb
visibilityTimeout: # [number; maximum: 43200] Visibility timeout seconds - After
numReceivers: # [number; minimum: 1, maximum: 100] Number of receivers - How many
socketTimeout: # [number; minimum: 1, maximum: 43200] Socket timeout - Socket ina
skipOnError: # [boolean] Skip file on error - Skip files that trigger a processin
includeSqsMetadata: # [boolean] Include notification metadata - Attach SQS notifi
enableAssumeRole: # [boolean] Enable for Amazon S3 - Use Assume Role credentials
assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the role
assumeRoleExternalId: # [string] External ID - External ID to use when assuming r
durationSeconds: # [number; minimum: 900, maximum: 43200] Duration (seconds) - Du
enableSQSAssumeRole: # [boolean] Enable for Amazon SQS - Use Assume Role credenti
preprocess: # [object]
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  command: # [string] Command - Command to feed the data through (via stdin) and
  args: # [array of strings] Arguments - Arguments to be added to the custom comm

  # -------------------------------------------------------

metadata: # [array] Fields - Fields to add to events from this input
```

```
  - name: # [string] Field Name
    value: # [string] Value - JavaScript expression to compute field's value, enc
parquetChunkSizeMB: # [number; minimum: 1, maximum: 100] Parquet chunk size limit
parquetChunkDownloadTimeout: # [number; minimum: 1, maximum: 3600] Parquet chunk
checkpointing: # [object]
  enabled: # [boolean] Enable checkpointing - Resume processing files after an ir

  # -------------- if enabled is true ---------------

  retries: # [number; maximum: 100] Retries - The number of times to retry proces

  # -------------------------------------------------------

pollTimeout: # [number; minimum: 1, maximum: 20] Poll timeout (secs) - How long t
type: # [string] Input Type
checksumSuffix: # [string] Checksum Suffix - Filename suffix of the manifest chec
maxManifestSizeKB: # [integer; minimum: 1] Manifest size limit (KB) - Maximum dow
validateInventoryFiles: # [boolean] Validate inventory files - If set to Yes, eac
description: # [string] Description
disabled: # [boolean] Disabled
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

# -------------- if sendToRoutes is false ---------------

connections: # [array] Use QuickConnect - Direct connections to Destinations, and
  - pipeline: # [string] Pipeline or Pack
    output: # [string] Destination

# -------------------------------------------------------

environment: # [string] Environment - Optionally, enable this config only on a sp
pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

# -------------- if pqEnabled is true ---------------

pq: # [object]
  mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
  maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
  commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
  maxFileSize: # [string] File size limit - The maximum size to store in each que
  maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
  path: # [string] Queue file path - The location for the persistent queue files.
```

```
    compress: # [string] Compression - Codec to use to compress the persisted data

  # ------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
snmp_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled
  host: # [string] Address - Address to bind on. For IPv4 (all addresses), use the
  port: # [number; maximum: 65535] [required] UDP port - UDP port to receive SNMP t
  snmpV3Auth: # [object] SNMPv3 authentication - Authentication parameters for SNMF
    v3AuthEnabled: # [boolean] Enabled

    # -------------- if v3AuthEnabled is true ---------------

    allowUnmatchedTrap: # [boolean] Allow unmatched traps - Pass through traps that
    v3Users: # [array] SNMP v3 users - User credentials for receiving v3 traps
      - name: # [string] V3 name
        authProtocol: # [string] Authentication protocol

    # -------------------------------------------------------

  maxBufferSize: # [number] Buffer size limit (events) - Maximum number of events t
  ipWhitelistRegex: # [string] IP allowlist regex - Regex matching IP addresses tha
  metadata: # [array] Fields - Fields to add to events from this input
    - name: # [string] Field Name
      value: # [string] Value - JavaScript expression to compute field's value, enc
  udpSocketRxBufSize: # [number; minimum: 256, maximum: 4294967295] UDP socket buff
  varbindsWithTypes: # [boolean] Include varbind types - If enabled, parses varbinc
  bestEffortParsing: # [boolean] Best effort parsing - If enabled, the parser will
  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Use QuickConnect - Direct connections to Destinations, and
    - pipeline: # [string] Pipeline or Pack
      output: # [string] Destination

  # -------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a sp
```

```
  pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # ---------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
open_telemetry_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled
  host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses
  port: # [number; maximum: 65535] [required] Port - Port to listen on
  protocol: # [string] Protocol - Select whether to leverage gRPC or HTTP for OpenT

  # -------------- if protocol is http ---------------

  tls: # [object] TLS settings (server side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    certificateName: # [string] Certificate - The name of the predefined certificat
    privKeyPath: # [string] Private key path - Path on server containing the privat
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
    certPath: # [string] Certificate path - Path on server containing certificates
    caPath: # [string] CA certificate path - Path on server containing CA certifica
    requestCert: # [boolean] Authenticate client (mutual auth) - Require clients to
    minVersion: # [string] Minimum TLS version
    maxVersion: # [string] Maximum TLS version

    # ---------------------------------------------------------

  maxActiveReq: # [number] Active request limit - Maximum number of active requests
  maxRequestsPerSocket: # [integer] Requests-per-socket limit - Maximum number of r
```

```
requestTimeout: # [number] Request timeout (seconds) - How long to wait for an ir
keepAliveTimeout: # [number; minimum: 1, maximum: 600] Keep-alive timeout (seconc
enableHealthCheck: # [boolean] Health check endpoint - Enable to expose the /crib
ipAllowlistRegex: # [string] IP allowlist regex - Messages from matched IP addres
ipDenylistRegex: # [string] IP denylist regex - Messages from matched IP addresse
socketTimeout: # [number] Socket timeout (seconds) - How long Cribl Edge should w

# ------------------------------------------------------


# -------------- if protocol is grpc ---------------

maxActiveCxn: # [number] Active connection limit - Maximum number of active conne
tls: # [object] TLS settings (server side)
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  certificateName: # [string] Certificate - The name of the predefined certificat
  privKeyPath: # [string] Private key path - Path on server containing the privat
  certPath: # [string] Certificate path - Path on server containing certificates
  caPath: # [string] CA certificate path - Path on server containing CA certifica
  requestCert: # [boolean] Authenticate client (mutual auth) - Require clients to
  minVersion: # [string] Minimum TLS version
  maxVersion: # [string] Maximum TLS version

  # ------------------------------------------------------


# ------------------------------------------------------

extractSpans: # [boolean] Extract spans - Enable to extract each incoming span to
extractMetrics: # [boolean] Extract metrics - Enable to extract each incoming Gau
otlpVersion: # [string] OTLP version - The version of OTLP Protobuf definitions t

# -------------- if otlpVersion is 1.3.1 ---------------

extractLogs: # [boolean] Extract logs - Enable to extract each incoming log recor

# ------------------------------------------------------

authType: # [string] Authentication type - OpenTelemetry authentication type
```

```
# -------------- if authType is basic ---------------

username: # [string] Username
password: # [string] Password

# ---------------------------------------------------


# -------------- if authType is token ---------------

token: # [string] Token - Bearer token to include in the authorization header

# ---------------------------------------------------


# -------------- if authType is credentialsSecret ---------------

credentialsSecret: # [string] Credentials secret - Select or create a secret that

# ---------------------------------------------------


# -------------- if authType is textSecret ---------------

textSecret: # [string] Token (text secret) - Select or create a stored text secre

# ---------------------------------------------------


# -------------- if authType is oauth ---------------

loginUrl: # [string] Login URL - URL for OAuth
secretParamName: # [string] OAuth Secret parameter name - Secret parameter name t
secret: # [string] OAuth secret - Secret parameter value to pass in request body
tokenAttributeName: # [string] Token attribute name - Name of the auth token attr
spacer: # [null]
authHeaderExpr: # [string] Authorize expression - JavaScript expression to comput
tokenTimeoutSecs: # [number; minimum: 1, maximum: 300000] Refresh interval (secs.
oauthParams: # [array] OAuth parameters - Additional parameters to send in the OA
  - name: # [string] Name - OAuth parameter name
    value: # [string] Value - OAuth parameter value
oauthHeaders: # [array] OAuth headers - Additional headers to send in the OAuth l
  - name: # [string] Name - OAuth header name
```

```
      value: # [string] Value - OAuth header value

  # ------------------------------------------------------

  metadata: # [array] Fields - Fields to add to events from this input
    - name: # [string] Field Name
      value: # [string] Value - JavaScript expression to compute field's value, enc
  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Use QuickConnect - Direct connections to Destinations, and
    - pipeline: # [string] Pipeline or Pack
      output: # [string] Destination

  # ------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a sp
  pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # ------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
model_driven_telemetry_input: # [object]
  type: # [string] Input Type
  host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses
  port: # [number; maximum: 65535] [required] Port - Port to listen on
  tls: # [object] TLS settings (server side)
    disabled: # [boolean] Disabled
```

```
  # -------------- if disabled is false ---------------

  certificateName: # [string] Certificate - The name of the predefined certificat
  privKeyPath: # [string] Private key path - Path on server containing the privat
  certPath: # [string] Certificate path - Path on server containing certificates
  caPath: # [string] CA certificate path - Path on server containing CA certifica
  requestCert: # [boolean] Authenticate client (mutual auth) - Require clients to
  minVersion: # [string] Minimum TLS version
  maxVersion: # [string] Maximum TLS version

  # --------------------------------------------------------

metadata: # [array] Fields - Fields to add to events from this input
  - name: # [string] Field Name
    value: # [string] Value - JavaScript expression to compute field's value, enc
maxActiveCxn: # [number] Active connection limit - Maximum number of active conne
shutdownTimeoutMs: # [number; minimum: 1] Shutdown timeout - Time in milliseconds
description: # [string] Description
disabled: # [boolean] Disabled
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

# -------------- if sendToRoutes is false ---------------

connections: # [array] Use QuickConnect - Direct connections to Destinations, and
  - pipeline: # [string] Pipeline or Pack
    output: # [string] Destination

# ---------------------------------------------------------

environment: # [string] Environment - Optionally, enable this config only on a sp
pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

# -------------- if pqEnabled is true ---------------

pq: # [object]
  mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
  maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
  commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
  maxFileSize: # [string] File size limit - The maximum size to store in each que
  maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
  path: # [string] Queue file path - The location for the persistent queue files.
  compress: # [string] Compression - Codec to use to compress the persisted data
```

```
  # ------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
sqs_input: # [object]
  type: # [string] Input Type
  queueName: # [string] Queue - The name, URL, or ARN of the SQS queue to read ever
  queueType: # [string] [required] Queue type - The queue type used (or created)

  # -------------- if queueType is standard ---------------

  numReceivers: # [number; minimum: 1, maximum: 100] Number of receivers - How many

  # ------------------------------------------------------

  awsAccountId: # [string] AWS account ID - SQS queue owner's AWS account ID. Leave
  createQueue: # [boolean] Create queue - Create queue if it does not exist
  awsAuthenticationMethod: # [string] Authentication method - AWS authentication me

  # -------------- if awsAuthenticationMethod is manual ---------------

  awsApiKey: # [string] Access key

  # ------------------------------------------------------


  # -------------- if awsAuthenticationMethod is secret ---------------

  awsSecret: # [string] Secret key pair - Select or create a stored secret that ref

  # ------------------------------------------------------

  awsSecretKey: # [string] Secret key
  region: # [string] Region - AWS Region where the SQS queue is located. Required,
  endpoint: # [string] Endpoint - SQS service endpoint. If empty, defaults to the A
  signatureVersion: # [string] Signature version - Signature version to use for sig
  reuseConnections: # [boolean] Reuse connections - Reuse connections between reque
  rejectUnauthorized: # [boolean] Reject unauthorized certificates - Reject certifi
  enableAssumeRole: # [boolean] Enable for SQS - Use Assume Role credentials to acc
  assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the role
  assumeRoleExternalId: # [string] External ID - External ID to use when assuming r
  durationSeconds: # [number; minimum: 900, maximum: 43200] Duration (seconds) - Du
  maxMessages: # [number; minimum: 1, maximum: 10] Message limit - The maximum numb
```

```
  visibilityTimeout: # [number; maximum: 43200] Visibility Timeout Seconds - After
  metadata: # [array] Fields - Fields to add to events from this input
    - name: # [string] Field Name
      value: # [string] Value - JavaScript expression to compute field's value, enc
  pollTimeout: # [number; minimum: 1, maximum: 20] Poll timeout (secs) - How long t
  description: # [string] Description
  disabled: # [boolean] Disabled
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Use QuickConnect - Direct connections to Destinations, and
    - pipeline: # [string] Pipeline or Pack
      output: # [string] Destination

  # ------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a sp
  pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # ------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
syslog_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled
  host: # [string] Address - Address to bind on. For IPv4 (all addresses), use the
  udpPort: # [number; maximum: 65535] UDP port - Enter UDP port number to listen or
  tcpPort: # [number; maximum: 65535] TCP port - Enter TCP port number to listen or
  maxBufferSize: # [number] Buffer size limit (events) - Maximum number of events t
  ipWhitelistRegex: # [string] IP allowlist regex - Regex matching IP addresses tha
```

```
timestampTimezone: # [string] Default timezone - Timezone to assign to timestamps
singleMsgUdpPackets: # [boolean] Single msg per UDP - Treat UDP packet data recei
enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection i

# -------------- if enableProxyHeader is true ---------------

enableEnhancedProxyHeaderParsing: # [boolean] Enable enhanced TLS handshake for p

# -----------------------------------------------------------

keepFieldsList: # [array of strings] Fields to keep - Wildcard list of fields to
octetCounting: # [boolean] Octet count framing - Enable if incoming messages use
inferFraming: # [boolean] Infer Syslog framing - Enable if we should infer the sy
strictlyInferOctetCounting: # [boolean] Strictly infer octet count framing - Enab
allowNonStandardAppName: # [boolean] Allow non-standard app name - Enable if RFC
maxActiveCxn: # [number] Active connection limit - Maximum number of active conne
socketIdleTimeout: # [number] TCP socket idle timeout (seconds) - How long Cribl
socketEndingMaxWait: # [number] TCP forced socket termination timeout (seconds) -
socketMaxLifespan: # [number] TCP Socket max lifespan (seconds) - The maximum dur
tls: # [object] TLS settings (server side)
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  certificateName: # [string] Certificate - The name of the predefined certificat
  privKeyPath: # [string] Private key path - Path on server containing the privat
  passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
  certPath: # [string] Certificate path - Path on server containing certificates
  caPath: # [string] CA certificate path - Path on server containing CA certifica
  requestCert: # [boolean] Authenticate client (mutual auth) - Require clients to
  minVersion: # [string] Minimum TLS version
  maxVersion: # [string] Maximum TLS version

  # -----------------------------------------------------------

metadata: # [array] Fields - Fields to add to events from this input
  - name: # [string] Field Name
    value: # [string] Value - JavaScript expression to compute field's value, enc
udpSocketRxBufSize: # [number; minimum: 256, maximum: 4294967295] UDP socket buff
enableLoadBalancing: # [boolean] Enable TCP load balancing - Load balance traffic
description: # [string] Description
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t
```

```
  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Use QuickConnect - Direct connections to Destinations, and
    - pipeline: # [string] Pipeline or Pack
      output: # [string] Destination

  # -------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a sp
  pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # -------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
file_input: # [object]
  type: # [string] Input Type
  mode: # [string]  - Choose how to discover files to monitor

  # -------------- if mode is manual ---------------

  path: # [string] Search path - Directory path to search for files. Environment va
  depth: # [number] Max depth - Set how many subdirectories deep to search. Use 0 t
  suppressMissingPathErrors: # [boolean] Suppress errors when search path does not
  deleteFiles: # [boolean] Delete files - Delete files after they have been collect

  # -------------------------------------------------------

  interval: # [number; minimum: 1] Polling interval - Time, in seconds, between sca
  filenames: # [array of strings] Filename allowlist - The full path of discovered
  tailOnly: # [boolean] Collect from end - Read only new entries at the end of all
  idleTimeout: # [number; minimum: 1] Idle timeout - Time, in seconds, before an ic
```

```
maxAgeDur: # [string] Age duration limit - The maximum age of files to monitor. F
checkFileModTime: # [boolean] Check file modification times - Skip files with mod
forceText: # [boolean] Force text format - Forces files containing binary data to

# -------------- if forceText is false ---------------

includeUnidentifiableBinary: # [boolean] Enable binary files - Stream binary file

# ----------------------------------------------------

hashLen: # [number; minimum: 1] Hash length - Length of file header bytes to use
metadata: # [array] Fields - Fields to add to events from this input
  - name: # [string] Field Name
    value: # [string] Value - JavaScript expression to compute field's value, enc
breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event-br
staleChannelFlushMs: # [number; minimum: 10, maximum: 43200000] Event Breaker buf
description: # [string] Description
disabled: # [boolean] Disabled
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

# -------------- if sendToRoutes is false ---------------

connections: # [array] Use QuickConnect - Direct connections to Destinations, and
  - pipeline: # [string] Pipeline or Pack
    output: # [string] Destination

# ----------------------------------------------------

environment: # [string] Environment - Optionally, enable this config only on a sp
pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

# -------------- if pqEnabled is true ---------------

pq: # [object]
  mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
  maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
  commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
  maxFileSize: # [string] File size limit - The maximum size to store in each que
  maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
  path: # [string] Queue file path - The location for the persistent queue files.
  compress: # [string] Compression - Codec to use to compress the persisted data
```

```
  # ------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
tcp_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled
  host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses
  port: # [number; maximum: 65535] [required] Port - Port to listen on
  tls: # [object] TLS settings (server side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    certificateName: # [string] Certificate - The name of the predefined certificat
    privKeyPath: # [string] Private key path - Path on server containing the privat
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
    certPath: # [string] Certificate path - Path on server containing certificates
    caPath: # [string] CA certificate path - Path on server containing CA certifica
    requestCert: # [boolean] Authenticate client (mutual auth) - Require clients to
    minVersion: # [string] Minimum TLS version
    maxVersion: # [string] Maximum TLS version

    # ------------------------------------------------------

  ipWhitelistRegex: # [string] IP allowlist regex - Regex matching IP addresses tha
  maxActiveCxn: # [number] Active connection limit - Maximum number of active conne
  socketIdleTimeout: # [number] Socket idle timeout (seconds) - How long Cribl Edge
  socketEndingMaxWait: # [number] Forced socket termination timeout (seconds) - How
  socketMaxLifespan: # [number] Socket max lifespan (seconds) - The maximum duratio
  enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection i
  metadata: # [array] Fields - Fields to add to events from this input
    - name: # [string] Field Name
      value: # [string] Value - JavaScript expression to compute field's value, enc
  breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event-br
  staleChannelFlushMs: # [number; minimum: 10, maximum: 43200000] Event Breaker buf
  enableHeader: # [boolean] Enable header - Client will pass the header record with

  # -------------- if enableHeader is true ---------------

  authType: # [string] Authentication method - Select Manual to enter an auth token

  # ------------------------------------------------------
```

```
  preprocess: # [object]
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    command: # [string] Command - Command to feed the data through (via stdin) and
    args: # [array of strings] Arguments - Arguments to be added to the custom comm

    # ----------------------------------------------------

  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Use QuickConnect - Direct connections to Destinations, and
    - pipeline: # [string] Pipeline or Pack
      output: # [string] Destination

  # -----------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a sp
  pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # -----------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
appscope_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled
  ipWhitelistRegex: # [string] IP allowlist regex - Regex matching IP addresses tha
```

```
maxActiveCxn: # [number] Active connection limit - Maximum number of active conne
socketIdleTimeout: # [number] Socket idle timeout (seconds) - How long Cribl Edge
socketEndingMaxWait: # [number] Forced socket termination timeout (seconds) - How
socketMaxLifespan: # [number] Socket max lifespan (seconds) - The maximum duratio
enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection i
metadata: # [array] Fields - Fields to add to events from this input
  - name: # [string] Field Name
    value: # [string] Value - JavaScript expression to compute field's value, enc
breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event-br
staleChannelFlushMs: # [number; minimum: 10, maximum: 43200000] Event Breaker buf
enableUnixPath: # [boolean] UNIX domain socket - Toggle to Yes to specify a file-

# -------------- if enableUnixPath is false ---------------

host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses
port: # [number; maximum: 65535] Port - Port to listen on
tls: # [object] TLS settings (server side)
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  certificateName: # [string] Certificate - The name of the predefined certificat
  privKeyPath: # [string] Private key path - Path on server containing the privat
  passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
  certPath: # [string] Certificate path - Path on server containing certificates
  caPath: # [string] CA certificate path - Path on server containing CA certifica
  requestCert: # [boolean] Authenticate client (mutual auth) - Require clients to
  minVersion: # [string] Minimum TLS version
  maxVersion: # [string] Maximum TLS version

  # ------------------------------------------------------


# ------------------------------------------------------


# -------------- if enableUnixPath is true ---------------

unixSocketPath: # [string] UNIX socket path - Path to the UNIX domain socket to l
unixSocketPerms: # [string,number] UNIX socket permissions - Permissions to set f

# ------------------------------------------------------
```

```
filter: # [object]
  allow: # [array] Rules - Specify processes that AppScope should be loaded into,
    - procname: # [string] Process name - Specify the name of a process or family
      arg: # [string] Process argument - Specify a string to substring-match agai
      config: # [string] AppScope config - Choose a config to apply to processes
  transportURL: # [string] Transport override - To override the UNIX domain socke
persistence: # [object] Persistence
  enable: # [boolean] Enable disk spooling - Spool events and metrics on disk for

  # -------------- if enable is true ---------------

  timeWindow: # [string] Bucket time span - Time span for each file bucket
  maxDataSize: # [string] Data size limit - Maximum disk space allowed to be cons
  maxDataTime: # [string] Data age limit - Maximum amount of time to retain data
  compress: # [string] Data compression format
  destPath: # [string] Path location - Path to use to write metrics. Defaults to

  # ------------------------------------------------------

authType: # [string] Authentication method - Select Manual to enter an auth token
authToken: # [string] Auth token - Shared secret to be provided by any client (ir

# -------------- if authType is manual ---------------


# ------------------------------------------------------

textSecret: # [string] Auth token (text secret) - Select or create a stored text

# -------------- if authType is secret ---------------


# ------------------------------------------------------

description: # [string] Description
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

# -------------- if sendToRoutes is false ---------------

connections: # [array] Use QuickConnect - Direct connections to Destinations, and
  - pipeline: # [string] Pipeline or Pack
    output: # [string] Destination
```

```
  # ------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a sp
  pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # ------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
wef_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled
  host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses
  port: # [number; maximum: 65535] [required] Port - Port to listen on
  authMethod: # [string] Authentication method - How to authenticate incoming clien

  # -------------- if authMethod is clientCert ---------------

  tls: # [object] mTLS settings
    disabled: # [boolean] Disabled - Enable TLS
    rejectUnauthorized: # [boolean] Validate client certs - Required for WEF certif
    requestCert: # [boolean] Authenticate client - Required for WEF certificate aut
    certificateName: # [string] Certificate - Name of the predefined certificate
    privKeyPath: # [string] Private key path - Path on server containing the privat
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
    certPath: # [string] [required] Certificate path - Path on server containing ce
    caPath: # [string] [required] CA certificate path - Server path containing CA c
    commonNameRegex: # [string] Common name - Regex matching allowable common names
    minVersion: # [string] Minimum TLS version
    maxVersion: # [string] Maximum TLS version
    ocspCheck: # [boolean] Verify certificate via OCSP - Enable OCSP check of certi
```

```
  # -------------- if ocspCheck is true ---------------

  ocspCheckFailClose: # [boolean] Strict validation - If enabled, checks will fai

  # ----------------------------------------------------------

caFingerprint: # [string] CA fingerprint override - SHA1 fingerprint expected by
allowMachineIdMismatch: # [boolean] Allow MachineID mismatch - Allow events to be
logFingerprintMismatch: # [boolean] Log CA fingerprint mismatch warning - Log a w

# ----------------------------------------------------------


# -------------- if authMethod is kerberos ---------------

keytab: # [string] Keytab location - Path to the keytab file containing the servi
principal: # [string] Service principal name - Kerberos principal used for auther

# ----------------------------------------------------------

maxActiveReq: # [number] Active request limit - Maximum number of active requests
maxRequestsPerSocket: # [integer] Requests-per-socket limit - Maximum number of r
enableProxyHeader: # [boolean] Show originating IP - Preserve the client's origir
captureHeaders: # [boolean] Capture request headers - Add request headers to ever
keepAliveTimeout: # [number; minimum: 1, maximum: 600] Keep-alive timeout (second
enableHealthCheck: # [boolean] Health check endpoint - Expose the /cribl_health e
ipAllowlistRegex: # [string] IP allowlist regex - Messages from matched IP addres
ipDenylistRegex: # [string] IP denylist regex - Messages from matched IP addresse
socketTimeout: # [number] Socket timeout (seconds) - How long Cribl Edge should w
subscriptions: # [array] [required] Subscriptions - Subscriptions to events on fo
  - subscriptionName: # [string] Subscription name
    version: # [string] Version - Version UUID for this subscription. If any subs
    contentFormat: # [string] Format - Content format in which the endpoint shoul
    heartbeatInterval: # [number; minimum: 1] Heartbeat - Maximum time (in second
    batchTimeout: # [number] Batch timeout - Interval (in seconds) over which the
    readExistingEvents: # [boolean] Read existing events - Newly subscribed endpo
    sendBookmarks: # [boolean] Use bookmarks - Keep track of which events have be
    compress: # [boolean] Compression - Receive compressed events from the source
    targets: # [array of strings] Targets - The DNS names of the endpoints that s
    locale: # [string] Locale - The RFC-3066 locale the Windows clients should us
    querySelector: # [string] Query builder mode
    metadata: # [array] Fields - Fields to add to events ingested under this subs
    - name: # [string] Field Name
```

```
      value: # [string] Value - JavaScript expression to compute field's value, e
  metadata: # [array] Fields - Fields to add to events from this input
    - name: # [string] Field Name
      value: # [string] Value - JavaScript expression to compute field's value, enc
  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Use QuickConnect - Direct connections to Destinations, and
    - pipeline: # [string] Pipeline or Pack
      output: # [string] Destination

  # --------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a sp
  pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # --------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
win_event_logs_input: # [object]
  type: # [string] Input Type
  logNames: # [array of strings] Event logs - Enter the event logs to collect. Run
  readMode: # [string] Read mode - Read all stored and future event logs, or only f
  eventFormat: # [string] Event format - Format of individual events
  disableNativeModule: # [boolean] Use Windows Tools - Enable to use built-in tools
  interval: # [number; minimum: 1] Polling interval - Time, in seconds, between che
  batchSize: # [number; minimum: 1] Batch size - The maximum number of events to re
  metadata: # [array] Fields - Fields to add to events from this input
    - name: # [string] Field Name
```

```
    value: # [string] Value - JavaScript expression to compute field's value, enc
  maxEventBytes: # [number; minimum: 1, maximum: 134217728] Max event bytes - The m
  description: # [string] Description
  disabled: # [boolean] Disabled
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Use QuickConnect - Direct connections to Destinations, and
    - pipeline: # [string] Pipeline or Pack
      output: # [string] Destination

  # --------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a sp
  pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # --------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
raw_udp_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled
  host: # [string] Address - Address to bind on. For IPv4 (all addresses), use the
  port: # [number; maximum: 65535] [required] Port - Port to listen on
  maxBufferSize: # [number] Buffer size limit (events) - Maximum number of events t
  ipWhitelistRegex: # [string] IP allowlist regex - Regex matching IP addresses tha
  singleMsgUdpPackets: # [boolean] Single msg per UDP - If true, each UDP packet is
  ingestRawBytes: # [boolean] Ingest raw bytes - If true, a __rawBytes field will b
  udpSocketRxBufSize: # [number; minimum: 256, maximum: 4294967295] UDP socket buff
  metadata: # [array] Fields - Fields to add to events from this input
```

```
    - name: # [string] Field Name
      value: # [string] Value - JavaScript expression to compute field's value, enc
  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Use QuickConnect - Direct connections to Destinations, and
    - pipeline: # [string] Pipeline or Pack
      output: # [string] Destination

  # --------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a sp
  pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # --------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
journal_files_input: # [object]
  type: # [string] Input Type
  path: # [string] Search path - Directory path to search for journals. Environment
  interval: # [number; minimum: 1] Polling interval - Time, in seconds, between sca
  journals: # [array of strings] [required] Journal allowlist - The full path of di
  rules: # [array] Filter Rules - Add rules to decide which journal objects to allo
    - filter: # [string] Filter Expression - JavaScript expression applied to Journ
      description: # [string] Description - Optional description of this rule's pur
  currentBoot: # [boolean] Current boot only - Skip log messages that are not part
  maxAgeDur: # [string] Age duration limit - The maximum log message age, in durati
  metadata: # [array] Fields - Fields to add to events from this input
    - name: # [string] Field Name
```

```
    value: # [string] Value - JavaScript expression to compute field's value, enc
  description: # [string] Description
  disabled: # [boolean] Disabled
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Use QuickConnect - Direct connections to Destinations, and
    - pipeline: # [string] Pipeline or Pack
      output: # [string] Destination

  # --------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a sp
  pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # --------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
wiz_input: # [object]
  type: # [string] Input Type
  endpoint: # [string] GraphQL endpoint - The Wiz GraphQL API endpoint. Example: ht
  authUrl: # [string] [required] Authentication URL - The authentication URL to ger
  authAudienceOverride: # [string] Authentication audience - The audience to use wh
  clientId: # [string] [required] Client ID - The client ID of the Wiz application
  contentConfig: # [array] [required] Content types
    - contentType: # [string] Content name - The name of the Wiz query
      contentDescription: # [string] Description
      enabled: # [boolean] Enable content
  requestTimeout: # [number; maximum: 2400] Request timeout (seconds) - HTTP reques
  keepAliveTime: # [number; minimum: 10] Keep alive time (seconds) - How often work
```

```
maxMissedKeepAlives: # [number; minimum: 2] Worker timeout (periods) - The numbe
ttl: # [string] Time to live - Time to keep the job's artifacts on disk after jo
ignoreGroupJobsLimit: # [boolean] Ignore Worker Group job limits - When enabled,
metadata: # [array] Fields - Fields to add to events from this input
  - name: # [string] Field Name
    value: # [string] Value - JavaScript expression to compute field's value, en
retryRules: # [object]
  type: # [string] Retry type - The algorithm to use when performing HTTP retrie

  # -------------- if type is static ---------------

  codes: # [array of numbers] Retry HTTP codes - List of HTTP codes that trigger
  interval: # [number; maximum: 20000] Wait (ms) - Time interval between failed r
  limit: # [number; maximum: 20] Retry limit - The maximum number of times to re
  enableHeader: # [boolean] Honor Retry-After header - Honor any Retry-After hea
  retryConnectTimeout: # [boolean] Retry connection timeout - Make a single retr
  retryConnectReset: # [boolean] Retry connection reset - Retry request when a c

  # ------------------------------------------------------


  # -------------- if type is backoff ---------------

  interval: # [number; maximum: 20000] Initial retry interval (ms) - Time interva
  limit: # [number; maximum: 20] Retry limit - The maximum number of times to re
  multiplier: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base for e
  codes: # [array of numbers] Retry HTTP codes - List of HTTP codes that trigger
  enableHeader: # [boolean] Honor Retry-After header - Honor any Retry-After hea
  retryConnectTimeout: # [boolean] Retry connection timeout - Make a single retr
  retryConnectReset: # [boolean] Retry connection reset - Retry request when a c

  # ------------------------------------------------------

authType: # [string] Authentication method - Enter client secret directly, or sel
clientSecret: # [string] Client secret - The client secret of the Wiz applicatio

# -------------- if authType is manual ---------------


# ------------------------------------------------------

textSecret: # [string] Client Secret (text secret) - Select or create a stored te
```

```
  # -------------- if authType is secret ---------------


  # -------------------------------------------------------

  description: # [string] Description
  disabled: # [boolean] Disabled
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Use QuickConnect - Direct connections to Destinations, and
    - pipeline: # [string] Pipeline or Pack
      output: # [string] Destination

  # -------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a sp
  pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # -------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
wiz_webhook_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled
  host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses
  port: # [number; maximum: 65535] [required] Port - Port to listen on
  authTokens: # [array of strings] Auth tokens - Shared secrets to be provided by a
  tls: # [object] TLS settings (server side)
    disabled: # [boolean] Disabled
```

```
  # -------------- if disabled is false ---------------

  certificateName: # [string] Certificate - The name of the predefined certificat
  privKeyPath: # [string] Private key path - Path on server containing the privat
  passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
  certPath: # [string] Certificate path - Path on server containing certificates
  caPath: # [string] CA certificate path - Path on server containing CA certifica
  requestCert: # [boolean] Authenticate client (mutual auth) - Require clients to
  minVersion: # [string] Minimum TLS version
  maxVersion: # [string] Maximum TLS version

  # -------------------------------------------------------

maxActiveReq: # [number] Active request limit - Maximum number of active requests
maxRequestsPerSocket: # [integer] Requests-per-socket limit - Maximum number of r
enableProxyHeader: # [boolean] Show originating IP - Extract the client IP and po
captureHeaders: # [boolean] Capture request headers - Add request headers to ever
activityLogSampleRate: # [number; minimum: 1] Activity log sample rate - How ofte
requestTimeout: # [number] Request timeout (seconds) - How long to wait for an ir
socketTimeout: # [number] Socket timeout (seconds) - How long Cribl Edge should w
keepAliveTimeout: # [number; minimum: 1, maximum: 600] Keep-alive timeout (second
enableHealthCheck: # [boolean] Health check endpoint - Expose the /cribl_health e
ipAllowlistRegex: # [string] IP allowlist regex - Messages from matched IP addres
ipDenylistRegex: # [string] IP denylist regex - Messages from matched IP addresse
breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event-br
staleChannelFlushMs: # [number; minimum: 10, maximum: 43200000] Event Breaker buf
metadata: # [array] Fields - Fields to add to events from this input
  - name: # [string] Field Name
    value: # [string] Value - JavaScript expression to compute field's value, enc
allowedPaths: # [array of strings] Allowed URI paths - List of URI paths accepted
allowedMethods: # [array of strings] Allowed HTTP methods - List of HTTP methods
authTokensExt: # [array] Auth tokens - Shared secrets to be provided by any clien
  - token: # [string] Token - Shared secret to be provided by any client (Authori
    description: # [string] Description
    metadata: # [array] Fields - Fields to add to events referencing this token
    - name: # [string] Field Name
      value: # [string] Value - JavaScript expression to compute field's value, e
description: # [string] Description
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

# -------------- if sendToRoutes is false ---------------
```

```
  connections: # [array] Use QuickConnect - Direct connections to Destinations, and
    - pipeline: # [string] Pipeline or Pack
      output: # [string] Destination

  # ------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a sp
  pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # ------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
netflow_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled
  host: # [string] Address - Address to bind on. For IPv4 (all addresses), use the
  port: # [number; maximum: 65535] [required] Port - Port to listen on
  enablePassThrough: # [boolean] Enable pass-through - Allow forwarding of events t
  ipAllowlistRegex: # [string] IP allowlist regex - Messages from matched IP addres
  ipDenylistRegex: # [string] IP denylist regex - Messages from matched IP addresse
  udpSocketRxBufSize: # [number; minimum: 256, maximum: 4294967295] UDP socket buff
  templateCacheMinutes: # [number; minimum: 1, maximum: 3600] Template cache minute
  v5Enabled: # [boolean] V5 - Accept messages in Netflow V5 format.
  v9Enabled: # [boolean] V9 - Accept messages in Netflow V9 format.
  ipfixEnabled: # [boolean] IPFIX - Accept messages in IPFIX format.
  metadata: # [array] Fields - Fields to add to events from this input
    - name: # [string] Field Name
      value: # [string] Value - JavaScript expression to compute field's value, enc
  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t
```

```
  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Use QuickConnect - Direct connections to Destinations, and
    - pipeline: # [string] Pipeline or Pack
      output: # [string] Destination

  # --------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a sp
  pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # --------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
security_lake_input: # [object]
  type: # [string] Input Type
  queueName: # [string] Queue - The name, URL, or ARN of the SQS queue to read noti
  fileFilter: # [string] Filename filter - Regex matching file names to download ar
  awsAccountId: # [string] AWS account ID - SQS queue owner's AWS account ID. Leave
  awsAuthenticationMethod: # [string] Authentication method - AWS authentication me

  # -------------- if awsAuthenticationMethod is manual ---------------

  awsApiKey: # [string] Access key

  # --------------------------------------------------------


  # -------------- if awsAuthenticationMethod is secret ---------------

  awsSecret: # [string] Secret key pair - Select or create a stored secret that ref
```

```
# ------------------------------------------------------

awsSecretKey: # [string] Secret key
region: # [string] Region - AWS Region where the S3 bucket and SQS queue are loca
endpoint: # [string] Endpoint - S3 service endpoint. If empty, defaults to the AW
signatureVersion: # [string] Signature version - Signature version to use for sig
reuseConnections: # [boolean] Reuse connections - Reuse connections between reque
rejectUnauthorized: # [boolean] Reject unauthorized certificates - Reject certifi
breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event-br
staleChannelFlushMs: # [number; minimum: 10, maximum: 43200000] Event Breaker buf
maxMessages: # [number; minimum: 1, maximum: 10] Message limit - The maximum numb
visibilityTimeout: # [number; maximum: 43200] Visibility timeout seconds - After
numReceivers: # [number; minimum: 1, maximum: 100] Number of receivers - How many
socketTimeout: # [number; minimum: 1, maximum: 43200] Socket timeout - Socket ina
skipOnError: # [boolean] Skip file on error - Skip files that trigger a processin
includeSqsMetadata: # [boolean] Include notification metadata - Attach SQS notifi
enableAssumeRole: # [boolean] Enable for Amazon S3 - Use Assume Role credentials
assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the role
assumeRoleExternalId: # [string] External ID - External ID to use when assuming r
durationSeconds: # [number; minimum: 900, maximum: 43200] Duration (seconds) - Du
enableSQSAssumeRole: # [boolean] Enable for Amazon SQS - Use Assume Role credenti
preprocess: # [object]
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  command: # [string] Command - Command to feed the data through (via stdin) and
  args: # [array of strings] Arguments - Arguments to be added to the custom comm

  # ------------------------------------------------------

metadata: # [array] Fields - Fields to add to events from this input
  - name: # [string] Field Name
    value: # [string] Value - JavaScript expression to compute field's value, enc
parquetChunkSizeMB: # [number; minimum: 1, maximum: 100] Parquet chunk size limit
parquetChunkDownloadTimeout: # [number; minimum: 1, maximum: 3600] Parquet chunk
checkpointing: # [object]
  enabled: # [boolean] Enable checkpointing - Resume processing files after an in

  # -------------- if enabled is true ---------------

  retries: # [number; maximum: 100] Retries - The number of times to retry proces
```

```
    # ------------------------------------------------------

  pollTimeout: # [number; minimum: 1, maximum: 20] Poll timeout (secs) - How long t
  encoding: # [string] Encoding - Character encoding to use when parsing ingested d
  description: # [string] Description
  disabled: # [boolean] Disabled
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly t

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Use QuickConnect - Direct connections to Destinations, and
    - pipeline: # [string] Pipeline or Pack
      output: # [string] Destination

  # -------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a sp
  pqEnabled: # [boolean] Enable persistent queue - Use a disk queue to minimize dat

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
    maxBufferSize: # [number; minimum: 42] Buffer size limit - The maximum number o
    commitFrequency: # [number; minimum: 1] Commit frequency - The number of events
    maxFileSize: # [string] File size limit - The maximum size to store in each que
    maxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    path: # [string] Queue file path - The location for the persistent queue files.
    compress: # [string] Compression - Codec to use to compress the persisted data

  # -------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
zscaler_hec_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled
  host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses
  port: # [number; maximum: 65535] [required] Port - Port to listen on
  authTokens: # [array] Auth tokens - Shared secrets to be provided by any client (
    - authType: # [string] Authentication method - Select Manual to enter an auth t
      enabled: # [boolean] Enable token
```

```
    description: # [string] Description
    allowedIndexesAtToken: # [array of strings] Allowed indexes - Enter the value
    metadata: # [array] Fields - Fields to add to events referencing this token
    - name: # [string] Field Name
      value: # [string] Value - JavaScript expression to compute field's value, e
tls: # [object] TLS settings (server side)
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  certificateName: # [string] Certificate - The name of the predefined certificat
  privKeyPath: # [string] Private key path - Path on server containing the privat
  passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
  certPath: # [string] Certificate path - Path on server containing certificates
  caPath: # [string] CA certificate path - Path on server containing CA certifica
  requestCert: # [boolean] Authenticate client (mutual auth) - Require clients to
  minVersion: # [string] Minimum TLS version
  maxVersion: # [string] Maximum TLS version

  # ------------------------------------------------------

maxActiveReq: # [number] Active request limit - Maximum number of active requests
maxRequestsPerSocket: # [integer] Requests-per-socket limit - Maximum number of r
enableProxyHeader: # [boolean] Show originating IP - Extract the client IP and po
captureHeaders: # [boolean] Capture request headers - Add request headers to ever
activityLogSampleRate: # [number; minimum: 1] Activity log sample rate - How ofte
requestTimeout: # [number] Request timeout (seconds) - How long to wait for an ir
socketTimeout: # [number] Socket timeout (seconds) - How long Cribl Edge should w
keepAliveTimeout: # [number; minimum: 1, maximum: 600] Keep-alive timeout (second
ipAllowlistRegex: # [string] IP allowlist regex - Messages from matched IP addres
ipDenylistRegex: # [string] IP denylist regex - Messages from matched IP addresse
hecAPI: # [string] [required] HEC endpoint - Absolute path on which to listen for
metadata: # [array] Fields - Fields to add to every event. May be overridden by f
  - name: # [string] Field Name
    value: # [string] Value - JavaScript expression to compute field's value, enc
allowedIndexes: # [array of strings] Allowed indexes - List values allowed in HEC
hecAcks: # [boolean] Zscaler HEC Acks - Whether to enable Zscaler HEC acknowledge
accessControlAllowOrigin: # [array of strings] CORS allowed origins - Optionally,
accessControlAllowHeaders: # [array of strings] CORS allowed headers - Optionally
emitTokenMetrics: # [boolean] Emit per-token request metrics - Enable to emit per
```

17.5.6. Instance.Yml

Instance configuration is located under `$CRIBL_HOME/local/_system/instance.yml` (`C:\ProgramData\Cribl\local\system\instance.yml` for Cribl Edge on Windows).

$CRIBL_HOME/local/_system/instance.yml or C:\ProgramData\Cribl\local_system.yml for Cribl Edge on Windows

```
distributed:
  # Mode - Choose one.
  # One of: single | worker | edge | managed-edge | master | outpost | search-supervi
  # [string; default: single; required]
  mode:
  # Deployment ID - Deployment ID for reporting telemetry on multiple deployments.
  # [string]
  reportedDeploymentId:
  # Default Worker Group - Worker Group to report to leader
  # [string; default: default]
  group:
  # Tags - Tag values to report to leader
  tags:
  # Env Var Regex - Regex to select environment variables to report to leader
  # [string; default: /^CRIBL_/]
  envRegex:
  master:
    # Address - Address to bind on. Defaults to 0.0.0.0 (all addresses).
    # [string; default: 0.0.0.0; required]
    host:
    # Port - Port to listen on
    # [number; max: 65535; default: 4200; required]
    port:
    # Forward to Leader API - Respond to API requests from the distributed port. True
    # [boolean; default: true]
    forwardToLeaderApi:
    # Auth token - Authentication token for Worker Nodes to communicate with Leader.
    # [string; default: criblmaster]
    authToken:
    # IP allowlist regex - Regex matching IP addresses that are allowed to establish
    # [string; default: /.*/]
    ipWhitelistRegex:
    # Active connection limit - Maximum number of active connections allowed from Wor
    # Use 0 for unlimited.
    # [number; min: 0; default: 0]
    maxActiveCxn:
    # Helper processes socket dir - Directory to hold sockets for inter-process commu
    # (IPC) between Leader and processes like Config Helpers and services. Defaults t
    # system's temp directory.
    # [string]
    configHelperSocketDir:
    # Resiliency - Enable or disable failover.
```

```
# One of: none | failover
# [string; default: none]
resiliency:
configBundles:
  # S3 Bundle Bucket URL - Bucket to use for remote bundle storage, in s3://${buc
  # [string]
  remoteUrl:
failover:
  # Failover volume - NFS (network file system) directory to support Leader failc
  # [string]
  volume:
  # Lease refresh period - How often the primary leader will refresh its hold on
  # file
  # [string; default: 5s]
  period:
  # Missed refresh limit - How many Lease refresh periods that can be missed befc
  # standby nodes attempt to elect themselves to primary
  # [number; default: 3]
  missedHBLimit:
tls:
  # Disabled
  # [boolean; default: true; required]
  disabled:
  # Certificate name - The name of the predefined certificate
  # [string]
  certificateName:
  # Private key path - Path on server in which to find the private key to use. PE
  # reference $ENV_VARS.
  # [string; required]
  privKeyPath:
  # Passphrase - Passphrase to use to decrypt private key
  # [string]
  passphrase:
  # Certificate path - Path on server in which to find certificates to use. PEM f
  # reference $ENV_VARS.
  # [string; required]
  certPath:
  # CA certificate path - Path on server where to find CA certificates to use. PE
  # reference $ENV_VARS.
  # [string]
  caPath:
  # Minimum TLS version - Minimum TLS version. Defaults to TLS 1.2.
  # One of: TLSv1 | TLSv1.1 | TLSv1.2 | TLSv1.3
```

```
  # [string]
  minVersion:
  # Maximum TLS version - Maximum TLS version. Defaults to TLS 1.3.
  # One of: TLSv1 | TLSv1.1 | TLSv1.2 | TLSv1.3
  # [string]
  maxVersion:
  # Default cipher list - Default suite of enabled and disabled TLS ciphers. Defa
  # ECDHE-RSA-AES128-GCM-SHA256:
  # ECDHE-ECDSA-AES128-GCM-SHA256:
  # ECDHE-RSA-AES256-GCM-SHA384:
  # ECDHE-ECDSA-AES256-GCM-SHA384:
  # DHE-RSA-AES128-GCM-SHA256:
  # ECDHE-RSA-AES128-SHA256:
  # DHE-RSA-AES128-SHA256:
  # ECDHE-RSA-AES256-SHA384:
  # DHE-RSA-AES256-SHA384:
  # ECDHE-RSA-AES256-SHA256:
  # DHE-RSA-AES256-SHA256:
  # HIGH:
  # !aNULL:
  # !eNULL:
  # !EXPORT:
  # !DES:
  # !RC4:
  # !MD5:
  # !PSK:
  # !SRP:
  # !CAMELLIA
  # [string]
  defaultCipherList:
  # ECDH curve - The curve name, or a colon-separated list of curve NIDs or names
  # ECDH key agreement. For example: 'P-521:P-384:P-256'. Defaults to 'auto'.
  # [string; default: auto]
  defaultEcdhCurve:
  # Validate server certs - Validate server certificates globally. Disable to all
  # self-signed certificates.
  # [boolean; default: true]
  rejectUnauthorized:
# Disable SNI-based connection routing - This is an advanced setting - do not tou
# under supervision of Cribl Support. This affects how connections are routed wit
# Control Plane. Changing this setting could affect the scalability of your syste
# [boolean; default: false]
disableSNIRouting:
```

```
# Compression - Codec to use to compress the data before sending
# One of: none | gzip
# [string; default: gzip]
compression:
# Connection timeout (ms) - Amount of time (milliseconds) to wait before assuming
# connection has failed
# [number; min: 0; default: 10000]
connectionTimeout:
# Write timeout (ms) - Amount of time (milliseconds) to wait for a write to compl
# assuming connection has failed
# [number; min: 0; default: 60000]
writeTimeout:
# Protocol
# One of: tcp | http2
# [string; default: tcp]
protocol:
proxy:
  # Disabled - Whether to proxy Leader comms through SOCKS
  # [boolean; default: true; required]
  disabled:
  # User ID - Username for SOCKS proxy authentication
  # [string]
  userId:
  # Password - Password for SOCKS proxy authentication
  # [string]
  password:
  # Port - SOCKS proxy port
  # [number; default: 1080]
```

17.5.7. JOBS.YML

`jobs.yml` maintains parameters for configured Collectors, corresponding to those listed on the **Collectors** page. Each collection job is listed according to the pattern shown below.

jobs.yml

```
collection_job: # [object]
  workerAffinity: # [boolean] Worker affinity - If enabled, tasks are created and run
  collector: # [object]
    type: # [string] Collector type - The type of collector to run

    # -------------- if type is azure_blob ---------------

    conf: # [object] [required]
      type: # [string]
      outputName: # [string] Auto-populate from - An optional predefined Destination
      authType: # [string] Authentication method - Enter authentication data directly

      # -------------- if authType is manual ---------------

      connectionString: # [string] Connection string - Enter your Azure storage accou

      # -----------------------------------------------------


      # -------------- if authType is secret ---------------

      textSecret: # [string] Connection string (text secret) - Text secret

      # -----------------------------------------------------


      # -------------- if authType is clientSecret ---------------

      storageAccountName: # [string] Storage account name - The name of your Azure st
      tenantId: # [string] Tenant ID - The service principal's tenant ID
      clientId: # [string] Client ID - The service principal's client ID
      clientTextSecret: # [string] Client secret (text secret) - Text secret containi
      endpointSuffix: # [string] Endpoint suffix - The endpoint suffix for the servic
      azureCloud: # [string] Azure Cloud - The Azure cloud to use. Defaults to Azure

      # -----------------------------------------------------


      # -------------- if authType is clientCert ---------------

      storageAccountName: # [string] Storage account name - The name of your Azure st
      tenantId: # [string] Tenant ID - The service principal's tenant ID
```

```
  clientId: # [string] Client ID - The service principal's client ID
  certificate: # [object]
    certificateName: # [string] Certificate - The certificate you registered as c
  azureCloud: # [string] Azure Cloud - The Azure cloud to use. Defaults to Azure
  endpointSuffix: # [string] Endpoint suffix - The endpoint suffix for the servic

  # ------------------------------------------------------

  containerName: # [string] Container name - Container to collect from. This valu
  path: # [string] Path - The directory from which to collect data. Templating is
  extractors: # [array] Path extractors - Extractors allow use of template tokens
    - key: # [string] Token - A token from the template path, such as epoch
      expression: # [string] Extractor Expression - A JavaScript expression that
  recurse: # [boolean] Recursive - Recurse through subdirectories
  includeMetadata: # [boolean] Include metadata - Include Azure Blob metadata in
  includeTags: # [boolean] Include tags - Include Azure Blob tags in collected ev
  maxBatchSize: # [number; minimum: 1] Batch size limit - Maximum number of metad
  parquetChunkSizeMB: # [number; minimum: 1, maximum: 100] Parquet chunk size lim
  parquetChunkDownloadTimeout: # [number; minimum: 1, maximum: 3600] Parquet chun

# ------------------------------------------------------


# -------------- if type is cribl_lake ---------------

conf: # [object] [required]
  type: # [string]
  dataset: # [string] Lake dataset - Lake dataset to collect data from.

# ------------------------------------------------------


# -------------- if type is database ---------------

conf: # [object] [required]
  type: # [string]
  connectionId: # [string] Connection - Select an existing Connection, or go to K
  query: # [string] [required] SQL Query - An expression that resolves to the que
  queryValidationEnabled: # [boolean] Validate Query - Enforces a basic query val
  defaultBreakers: # [string] Hidden Default Breakers
  __scheduling: # [object]
    stateTracking: # [object]
      enabled: # [boolean] Enabled - Enable tracking of collection progress betwe
```

```
      # -------------- if enabled is true ---------------

      trackingColumn: # [string] Tracking Column - A numeric database column whos

      # ------------------------------------------------------


# ------------------------------------------------------


# -------------- if type is filesystem ---------------

conf: # [object] [required]
  type: # [string]
  outputName: # [string] Auto-populate from - Select a predefined configuration (
  path: # [string] Directory - The directory from which to collect data. Templati
  extractors: # [array] Path extractors - Allows using template tokens as context
    - key: # [string] Token - A token from the template directory, such as epoch
      expression: # [string] Extractor expression - JavaScript expression that re
  recurse: # [boolean] Recursive - Recurse through subdirectories
  maxBatchSize: # [number; minimum: 1] Batch size limit (files) - Maximum number

# ------------------------------------------------------


# -------------- if type is google_cloud_storage ---------------

conf: # [object] [required]
  type: # [string]
  outputName: # [string] Auto-populate from - Name of the predefined Destination
  bucket: # [string] Bucket name - Name of the bucket to collect from. This value
  path: # [string] Path - The directory from which to collect data. Templating is
  extractors: # [array] Path extractors - Allows using template tokens as context
    - key: # [string] Token - A token from the template path, such as epoch
      expression: # [string] Extractor Expression - JavaScript expression that re
  endpoint: # [string] Endpoint - Google Cloud Storage service endpoint. If empty
  disableTimeFilter: # [boolean] Disable time filter - Used to disable Collector
  recurse: # [boolean] Recursive - Recurse through subdirectories
  maxBatchSize: # [number; minimum: 1] Batch size limit (objects) - Maximum numbe
  authType: # [string] Authentication method - Enter account credentials manually

  # -------------- if authType is manual ---------------
```

```
  serviceAccountCredentials: # [string] Service account credentials - Contents of

  # ------------------------------------------------------


  # -------------- if authType is secret ---------------

  textSecret: # [string] Service account credentials (text secret) - Select or cr

  # ------------------------------------------------------

  parquetChunkSizeMB: # [number; minimum: 1, maximum: 100] Parquet chunk size lim
  parquetChunkDownloadTimeout: # [number; minimum: 1, maximum: 3600] Parquet chur

# ------------------------------------------------------


# -------------- if type is health_check ---------------

conf: # [object] [required]
  discovery: # [object]
    discoverType: # [string] Discover Type - Defines how task discovery will be p

    # -------------- if discoverType is http ---------------

    discoverUrl: # [string] Discover URL - Expression to derive URL to use for th
    discoverMethod: # [string] Discover method - Discover HTTP method.
    discoverRequestHeaders: # [array] Discover Headers - Optional discover reques
      - name: # [string] Name - Header name.
        value: # [string] Value - JavaScript expression to compute the header val
    discoverDataField: # [string] Discover Data Field - Path to field in the resp

    # ------------------------------------------------------


    # -------------- if discoverType is json ---------------

    manualDiscoverResult: # [string] Discover result - Allows hard-coding the Dis
    discoverDataField: # [string] Discover data field - Within the response JSON,

    # ------------------------------------------------------
```

```
  # -------------- if discoverType is list ---------------

  itemList: # [array of strings] Discover items - Comma-separated list of items

  # ------------------------------------------------------

collectUrl: # [string] Health check URL - Expression to derive URL to use for t
collectMethod: # [string] [required] Health check method - Health check HTTP me

# -------------- if collectMethod is get ---------------

collectRequestParams: # [array] Health check parameters - Optional health check
  - name: # [string] Name - Parameter name
    value: # [string] Value - JavaScript expression to compute the parameter va

# ------------------------------------------------------


# -------------- if collectMethod is post ---------------

collectRequestParams: # [array] Health check parameters - Optional health check
  - name: # [string] Name - Parameter name.
    value: # [string] Value - JavaScript expression to compute the parameter va

# ------------------------------------------------------


# -------------- if collectMethod is post_with_body ---------------

collectBody: # [string] Health check POST Body - Template for POST body to send

# ------------------------------------------------------

collectRequestHeaders: # [array] Health check headers - Optional health check r
  - name: # [string] Name - Header Name
    value: # [string] Value - JavaScript expression to compute the header value
authenticateCollect: # [boolean] Authenticate health check - Enable to make aut
authentication: # [string] [required] Authentication - Authentication method fo

# -------------- if authentication is basic ---------------

username: # [string] Username - Basic authentication username
```

```
password: # [string] Password - Basic authentication password

# ------------------------------------------------------


# -------------- if authentication is basicSecret ---------------

credentialsSecret: # [string] Credentials secret - Select or create a stored se

# ------------------------------------------------------


# -------------- if authentication is login ---------------

loginUrl: # [string] Login URL - URL to use for login API call. This call is ex
username: # [string] Username - Login username
password: # [string] Password - Login password
loginBody: # [string] POST body - Template for POST body to send with login req
tokenRespAttribute: # [string] Token attribute - Path to token attribute in log
authHeaderExpr: # [string] Authorize Expression - JavaScript expression to comp
authRequestHeaders: # [array] Authentication Headers - Optional authentication
  - name: # [string] Name - Header name.
    value: # [string] Value - JavaScript expression to compute the header value

# ------------------------------------------------------


# -------------- if authentication is loginSecret ---------------

loginUrl: # [string] Login URL - URL to use for login API call, this call is ex
credentialsSecret: # [string] Credentials secret - Select or create a stored se
loginBody: # [string] POST body - Template for POST body to send with login req
tokenRespAttribute: # [string] Token attribute - Path to token attribute in log
authHeaderExpr: # [string] Authorize Expression - JavaScript expression to comp
authRequestHeaders: # [array] Authentication Headers - Optional authentication
  - name: # [string] Name - Header name.
    value: # [string] Value - JavaScript expression to compute the header value

# ------------------------------------------------------


# -------------- if authentication is oauth ---------------
```

```
loginUrl: # [string] Login URL - URL to use for the OAuth API call. This call i
tokenRespAttribute: # [string] Token attribute - Path to token attribute in log
authHeaderExpr: # [string] Authorize expression - JavaScript expression to comp
clientSecretParamName: # [string] Client secret parameter - Parameter name that
clientSecretParamValue: # [string] Client secret value - Secret value to add to
authRequestParams: # [array] Extra authentication parameters - OAuth request pa
  - name: # [string] Name - Parameter name.
    value: # [string] Value - JavaScript expression to compute the parameter's
authRequestHeaders: # [array] Authentication headers - Optional authentication
  - name: # [string] Name - Header name.
    value: # [string] Value - JavaScript expression to compute the header's val

# -------------------------------------------------------


# -------------- if authentication is oauthSecret ---------------

loginUrl: # [string] Login URL - URL to use for the OAuth API call. This call i
tokenRespAttribute: # [string] Token attribute - Path to token attribute in log
authHeaderExpr: # [string] Authorize expression - JavaScript expression to comp
clientSecretParamName: # [string] Client secret parameter - Parameter name that
textSecret: # [string] Client secret value (text secret) - Select or create a t
authRequestParams: # [array] Extra authentication parameters - OAuth request pa
  - name: # [string] Name - Parameter name.
    value: # [string] Value - JavaScript expression to compute the parameter's
authRequestHeaders: # [array] Authentication headers - Optional authentication
  - name: # [string] Name - Header name.
    value: # [string] Value - JavaScript expression to compute the header's val

# -------------------------------------------------------

timeout: # [number; maximum: 1800] Request Timeout (secs) - HTTP request inacti
rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to r
defaultBreakers: # [string] Hidden Default Breakers
safeHeaders: # [array of strings] Safe headers - List of headers that are safe
retryRules: # [object]
  type: # [string] Retry type - The algorithm to use when performing HTTP retri

  # -------------- if type is static ---------------

  interval: # [number; maximum: 20000] Wait (ms) - Time interval between retrie
  limit: # [number; maximum: 20] Retry limit - The maximum number of times to r
  codes: # [array of numbers] Retry HTTP codes - List of HTTP codes that trigge
```

```
    enableHeader: # [boolean] Honor Retry-After header - Honor any Retry-After he

    # ---------------------------------------------------------


    # -------------- if type is backoff ---------------

    interval: # [number; maximum: 20000] Initial retry interval (ms) - Time inter
    limit: # [number; maximum: 20] Retry limit - The maximum number of times to r
    multiplier: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base for
    codes: # [array of numbers] Retry HTTP codes - List of HTTP codes that trigge
    enableHeader: # [boolean] Honor Retry-After header - Honor any Retry-After he

    # ---------------------------------------------------------


# ---------------------------------------------------------


# -------------- if type is office365_mgmt ---------------

conf: # [object] [required]
  plan_type: # [string] Subscription plan - Office 365 subscription plan for your
  tenant_id: # [string] [required] Tenant identifier - Directory ID (tenant ident
  app_id: # [string] [required] Application identifier - Identifier of the regist
  client_secret: # [string] [required] Client secret - Application key of the reg
  publisher_identifier: # [string] Publisher identifier - Optional PublisherIdent
  content_type: # [string] [required] Content type - The type of content to retri
  ingestionLag: # [number] Ingestion lag (minutes) - This is necessary because th

# ---------------------------------------------------------


# -------------- if type is office365_service ---------------

conf: # [object] [required]
  tenant_id: # [string] Tenant identifier - Directory ID (tenant identifier) in A
  app_id: # [string] [required] Application identifier - Identifier of the regist
  client_secret: # [string] [required] Client secret - Application key of the reg
  content_type: # [string] [required] Content type - The type of content to retri

# ---------------------------------------------------------
```

```
# -------------- if type is prometheus ---------------

conf: # [object] [required]
  dimensionList: # [array] Extra Dimensions - Other dimensions to include in ever
  username: # [string] Username - Optional username for Basic authentication
  password: # [string] Password - Optional password for Basic authentication
  discoveryType: # [string] Discovery Type - Target discovery mechanism, use stat

  # -------------- if discoveryType is static ---------------

  targetList: # [array] Targets - List of Prometheus targets to pull metrics from

  # ----------------------------------------------------------


  # -------------- if discoveryType is dns ---------------

  nameList: # [array] DNS Names - List of DNS names to resolve
  recordType: # [string] Record Type - DNS Record type to resolve
  scrapeProtocol: # [string] Metrics Protocol - Protocol to use when collecting m
  scrapePath: # [string] Metrics Path - Path to use when collecting metrics from

  # ----------------------------------------------------------


  # -------------- if discoveryType is ec2 ---------------

  usePublicIp: # [boolean] Use Public IP - Use public IP address for discovered t
  scrapeProtocol: # [string] Metrics Protocol - Protocol to use when collecting m
  scrapePort: # [number; minimum: 1, maximum: 65535] Metrics Port - The port numb
  scrapePath: # [string] Metrics Path - Path to use when collecting metrics from
  searchFilter: # [array] Search Filter - EC2 Instance Search Filter
    - Name: # [string] Filter Name - Search filter attribute name, see: https://d
      Values: # [array of strings] Filter Values - Search Filter Values
  region: # [string] Region - Region from which to retrieve data.
  awsAuthenticationMethod: # [string] Authentication method - AWS authentication
  enableAssumeRole: # [boolean] Enable Assume Role - Use Assume Role credentials
  endpoint: # [string] Endpoint - EC2 service endpoint. If empty, defaults to AWS
  signatureVersion: # [string] Signature version - Signature version to use for s

  # ----------------------------------------------------------
```

```
# ------------------------------------------------------


# -------------- if type is rest ---------------

conf: # [object] [required]
  type: # [string]
  discovery: # [object]
    discoverType: # [string] Discover type - Defines how task discovery will be p

    # -------------- if discoverType is http ---------------

    discoverUrl: # [string] Discover URL - URL to use for the Discover operation.
    discoverMethod: # [string] Discover method
    discoverRequestHeaders: # [array] Discover headers
      - name: # [string] Name
        value: # [string] Value - JavaScript expression to compute parameter valu
    pagination: # [object]
      type: # [string] Pagination

      # -------------- if type is response_body ---------------

      attribute: # [array,string] Response attributes - Names of attributes withi
      maxPages: # [number] Page limit - Maximum number of pages to retrieve for t
      lastPageExpr: # [string] Last-page expression - JavaScript expression used

      # ------------------------------------------------------


      # -------------- if type is response_header ---------------

      attribute: # [array,string] Response attributes - Names of attributes withi
      maxPages: # [number] Page limit - Maximum number of pages to retrieve for t

      # ------------------------------------------------------


      # -------------- if type is response_header_link ---------------

      nextRelationAttribute: # [string] Next page relation name - Relation name u
      curRelationAttribute: # [string] Current page relation name - Relation name
      maxPages: # [number] Page limit - Maximum number of pages to retrieve for t
```

```
  # ------------------------------------------------------


  # -------------- if type is request_offset ---------------

  offsetField: # [string] Offset field name - Query string parameter that set
  offset: # [number] Starting offset - Offset index from which to start reque
  offsetSpacer: # [null]
  limitField: # [string] Limit field name - Query string parameter that sets
  limit: # [number; minimum: 1] Record limit - Maximum number of records to r
  limitSpacer: # [null]
  totalRecordField: # [string] Total record count field name - Name of the at
  maxPages: # [number] Page limit - Maximum number of pages to retrieve for t
  zeroIndexed: # [boolean] Zero-based index - Enable to indicate that the fir

  # ------------------------------------------------------


  # -------------- if type is request_page ---------------

  pageField: # [string] Page number field name - Query string parameter that
  page: # [number] Starting page number - Page number from which to start rec
  offsetSpacer: # [null]
  sizeField: # [string] Page size field name - Query string parameter that se
  size: # [number; minimum: 1] Record limit - Maximum number of records to re
  limitSpacer: # [null]
  totalPageField: # [string] Total page count field name - Name of the attrib
  totalRecordField: # [string] Total record count field name - Name of the at
  maxPages: # [number] Page limit - Maximum number of pages to retrieve for t
  zeroIndexed: # [boolean] Zero-based index - Enable to indicate that the fir

  # ------------------------------------------------------

discoverDataField: # [string] Discover data field - Path to field in the resp
enableDiscoverCode: # [boolean] Format discover result with custom code

# ------------------------------------------------------


# -------------- if discoverType is json ---------------

manualDiscoverResult: # [string] Discover result - Allows hard-coding the Dis
```

```
  discoverDataField: # [string] Discover data field - Within the response JSON,

  # ------------------------------------------------------

  # -------------- if discoverType is list ---------------

  itemList: # [array of strings] Discover items - Comma-separated list of items

  # ------------------------------------------------------

collectUrl: # [string] Collect URL - URL (constant or JavaScript expression) to
collectMethod: # [string] [required] Collect method

# -------------- if collectMethod is get ---------------

collectRequestParams: # [array] Collect parameters
  - name: # [string] Name
    value: # [string] Value - JavaScript expression to compute parameter value,

# ------------------------------------------------------

# -------------- if collectMethod is post ---------------

collectRequestParams: # [array] Collect parameters
  - name: # [string] Name
    value: # [string] Value - JavaScript expression to compute parameter value,

# ------------------------------------------------------

# -------------- if collectMethod is post_with_body ---------------

collectBody: # [string] Collect POST body - Template for POST body to send with

# ------------------------------------------------------

# -------------- if collectMethod is other ---------------

collectVerb: # [string] Collect verb - Custom HTTP method to use for the Collec
collectBody: # [string] Collect body - Template for body to send with the Colle
```

```
collectRequestParams: # [array] Collect parameters
  - name: # [string] Name
    value: # [string] Value - JavaScript expression to compute parameter value,

# -----------------------------------------------------

collectRequestHeaders: # [array] Collect headers
  - name: # [string] Name
    value: # [string] Value - JavaScript expression to compute parameter value,
pagination: # [object]
  type: # [string] Pagination

  # -------------- if type is response_body ---------------

  attribute: # [array,string] Response attributes - Names of attributes within
  maxPages: # [number] Page limit - Maximum number of pages to retrieve per col
  lastPageExpr: # [string] Last-page expression - JavaScript expression used to

  # -----------------------------------------------------


  # -------------- if type is response_header ---------------

  attribute: # [array,string] Response attributes - Names of attributes within
  maxPages: # [number] Page limit - Maximum number of pages to retrieve per col

  # -----------------------------------------------------


  # -------------- if type is response_header_link ---------------

  nextRelationAttribute: # [string] Next page relation name - Relation name use
  curRelationAttribute: # [string] Current page relation name - Relation name u
  maxPages: # [number] Page limit - Maximum number of pages to retrieve per col

  # -----------------------------------------------------


  # -------------- if type is request_offset ---------------

  offsetField: # [string] Offset field name - Query string parameter that sets
  offset: # [number] Starting offset - Offset index from which to start request
  offsetSpacer: # [null]
```

```
  limitField: # [string] Limit field name - Query string parameter that sets th
  limit: # [number; minimum: 1] Record limit - Maximum number of records to col
  limitSpacer: # [null]
  totalRecordField: # [string] Total record count field name - Name of the attr
  maxPages: # [number] Page limit - Maximum number of pages to retrieve per col
  zeroIndexed: # [boolean] Zero-based index - Enable to indicate that the first

  # ------------------------------------------------------


  # -------------- if type is request_page ---------------

  pageField: # [string] Page number field name - Query string parameter that se
  page: # [number] Starting page number - Page number from which to start reque
  offsetSpacer: # [null]
  sizeField: # [string] Page size field name - Query string parameter that sets
  size: # [number; minimum: 1] Record limit - Maximum number of records to coll
  limitSpacer: # [null]
  totalPageField: # [string] Total page count field name - Name of the attribut
  totalRecordField: # [string] Total record count field name - Name of the attr
  maxPages: # [number] Page limit - Maximum number of pages to retrieve per col
  zeroIndexed: # [boolean] Zero-based index - Enable to indicate that the first

  # ------------------------------------------------------

authentication: # [string] [required] Authentication - Authentication method fo

# -------------- if authentication is basic ---------------

username: # [string] Username
password: # [string] Password

# ------------------------------------------------------


# -------------- if authentication is basicSecret ---------------

credentialsSecret: # [string] Credentials secret - Select or create a stored se

# ------------------------------------------------------


# -------------- if authentication is login ---------------
```

```
loginUrl: # [string] Login URL - URL to use for login API call. This call is ex
username: # [string] Login username
password: # [string] Login password
loginBody: # [string] POST body - Template for POST body to send with login req
tokenRespAttribute: # [string] Token attribute - Path to token attribute in log
authHeaderKey: # [string] Authorization header - Authorization header key to pa
authHeaderExpr: # [string] Authorize expression - JavaScript expression used to
authRequestHeaders: # [array] Authentication headers
  - name: # [string] Name
    value: # [string] Value - JavaScript expression to compute parameter value,

# -------------------------------------------------------


# -------------- if authentication is loginSecret ---------------

loginUrl: # [string] Login URL - URL to use for login API call. This call is ex
credentialsSecret: # [string] Credentials secret - Select or create a stored se
loginBody: # [string] POST body - Template for POST body to send with login req
tokenRespAttribute: # [string] Token attribute - Path to token attribute in log
authHeaderKey: # [string] Authorization header - Authorization header key to pa
authHeaderExpr: # [string] Authorize expression - JavaScript expression to comp
authRequestHeaders: # [array] Authentication headers
  - name: # [string] Name
    value: # [string] Value - JavaScript expression to compute parameter value,

# -------------------------------------------------------


# -------------- if authentication is oauth ---------------

loginUrl: # [string] Login URL - URL to use for the OAuth API call. This call i
tokenRespAttribute: # [string] Token attribute - Path to token attribute in log
authHeaderKey: # [string] Authorization header - Authorization header key to pa
authHeaderExpr: # [string] Authorize expression - JavaScript expression to comp
clientSecretParamName: # [string] Client secret parameter - Defaults to 'client
clientSecretParamValue: # [string] Client secret value - Secret value to add to
authRequestParams: # [array] Extra authentication parameters - OAuth request pa
  - name: # [string] Name
    value: # [string] Value - JavaScript expression to compute parameter value,
authRequestHeaders: # [array] Authentication headers
  - name: # [string] Name
```

```
    value: # [string] Value - JavaScript expression to compute parameter value,

# ------------------------------------------------------


# -------------- if authentication is oauthSecret ---------------

loginUrl: # [string] Login URL - URL to use for the OAuth API call. This call i
tokenRespAttribute: # [string] Token attribute - Path to token attribute in log
authHeaderKey: # [string] Authorization header - Authorization header key to pa
authHeaderExpr: # [string] Authorize expression - JavaScript expression to comp
clientSecretParamName: # [string] Client secret parameter - Defaults to 'client
textSecret: # [string] Client secret value (text secret) - Select or create a t
authRequestParams: # [array] Extra authentication parameters - OAuth request pa
  - name: # [string] Name
    value: # [string] Value - JavaScript expression to compute parameter value,
authRequestHeaders: # [array] Authentication headers
  - name: # [string] Name
    value: # [string] Value - JavaScript expression to compute parameter value,

# ------------------------------------------------------


# -------------- if authentication is google_oauth ---------------

scopes: # [array of strings] Scopes - Scopes to use during authentication. See
serviceAccountCredentials: # [string] Service account credentials - Contents of
subject: # [string] Impersonated account's email address - Email address of a u

# ------------------------------------------------------


# -------------- if authentication is google_oauthSecret ---------------

scopes: # [array of strings] Scopes - Scopes to use during authentication. See
textSecret: # [string] Service account credentials (text secret) - Select or cr
subject: # [string] Impersonated account's email address - Email address of a u

# ------------------------------------------------------


# -------------- if authentication is hmac ---------------
```

```
hmacFunctionId: # [string] HMAC Function - Select or create an HMAC Function to

# ------------------------------------------------------

timeout: # [number; maximum: 1800] Request timeout (secs) - HTTP request inacti
useRoundRobinDns: # [boolean] Round-robin DNS - Use round-robin DNS lookup. Sui
disableTimeFilter: # [boolean] Disable time filter - Disable Collector event ti
decodeUrl: # [boolean] Decode URL - Decode the URL before sending requests (inc
rejectUnauthorized: # [boolean] Reject unauthorized certificates - Reject certi
captureHeaders: # [boolean] Capture response headers - Enable to add response h
safeHeaders: # [array of strings] Safe headers - List of headers that are safe
retryRules: # [object]
  type: # [string] Retry type - Algorithm to use when performing HTTP retries

  # -------------- if type is static ---------------

  interval: # [number; maximum: 20000] Wait (ms) - Time interval between retrie
  limit: # [number; maximum: 20] Retry limit - Maximum number of times to retry
  codes: # [array of numbers] Retry HTTP codes - List of HTTP codes that trigge
  enableHeader: # [boolean] Honor Retry-After header - Honor any Retry-After he
  retryConnectTimeout: # [boolean] Retry connection timeout - Make a single ret
  retryConnectReset: # [boolean] Retry connection reset - Retry request when a

  # ------------------------------------------------------


  # -------------- if type is backoff ---------------

  interval: # [number; maximum: 20000] Initial retry interval (ms) - Time inter
  limit: # [number; maximum: 20] Retry limit - Maximum number of times to retry
  multiplier: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base for
  maxIntervalMs: # [number] Longest interval between retries (ms)
  codes: # [array of numbers] Retry HTTP codes - List of HTTP codes that trigge
  enableHeader: # [boolean] Honor Retry-After header - Honor any Retry-After he
  retryConnectTimeout: # [boolean] Retry connection timeout - Make a single ret
  retryConnectReset: # [boolean] Retry connection reset - Retry request when a

  # ------------------------------------------------------

__scheduling: # [object]
  stateTracking: # [object]
    enabled: # [boolean] Enabled - Track collection progress between consecutiv
```

```
      # -------------- if enabled is true ---------------

      stateUpdateExpression: # [string] State update expression - JavaScript expr
      stateMergeExpression: # [string] State merge expression - JavaScript expres

      # --------------------------------------------------------


# ----------------------------------------------------


# -------------- if type is s3 ---------------

conf: # [object] [required]
  type: # [string]
  outputName: # [string] Auto-populate from - Name of the predefined Destination
  bucket: # [string] S3 bucket - S3 Bucket from which to collect data
  parquetChunkSizeMB: # [number; minimum: 1, maximum: 100] Parquet chunk size lim
  parquetChunkDownloadTimeout: # [number; minimum: 1, maximum: 3600] Parquet chun
  region: # [string] Region - Region from which to retrieve data
  path: # [string] Path - Directory where data will be collected. Templating (suc
  partitioningScheme: # [string] Partitioning scheme - Partitioning scheme used f

  # -------------- if partitioningScheme is none ---------------

  recurse: # [boolean] Recursive - Traverse and include files from subdirectories

  # --------------------------------------------------------

  extractors: # [array] Path extractors - Allows using template tokens as context
    - key: # [string] Token - A token from the template path, such as epoch
      expression: # [string] Extractor Expression - JavaScript expression that re
  awsAuthenticationMethod: # [string] Authentication method - AWS authentication

  # -------------- if awsAuthenticationMethod is manual ---------------

  awsApiKey: # [string] Access key - Access key. If not present, will fall back t
  awsSecretKey: # [string] Secret key - Secret key. If not present, will fall bac

  # --------------------------------------------------------


  # -------------- if awsAuthenticationMethod is secret ---------------
```

```
  awsSecret: # [string] Secret key pair - Select or create a stored secret that r

  # ------------------------------------------------------

  endpoint: # [string] Endpoint - Must point to an S3-compatible endpoint. If emp
  signatureVersion: # [string] Signature version - Signature version to use for s
  enableAssumeRole: # [boolean] Enable Assume Role - Use AssumeRole credentials
  assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the ro
  assumeRoleExternalId: # [string] External ID - External ID to use when assuming
  durationSeconds: # [number; minimum: 900, maximum: 43200] Duration (seconds) -
  maxBatchSize: # [number; minimum: 1] Batch size limit (objects) - Maximum numbe
  reuseConnections: # [boolean] Reuse connections - Reuse connections between rec
  rejectUnauthorized: # [boolean] Reject unauthorized certificates - Reject certi
  verifyPermissions: # [boolean] Verify bucket permissions - Disable if you can a
  disableTimeFilter: # [boolean] Disable time filter - Disable Collector event ti

# ------------------------------------------------------


# -------------- if type is script ---------------

conf: # [object] [required]
  type: # [string]
  discoverScript: # [string] Discover Script - Script to discover what to collect
  collectScript: # [string] [required] Collect Script - Script to run to perform
  shell: # [string] Shell - Shell to use to execute scripts.
  envVars: # [array] Environment Variables - Environment variables to expose to t
    - name: # [string] Name - Environment variable name
      value: # [string] Value - JavaScript expression to compute environment vari

# ------------------------------------------------------


# -------------- if type is splunk ---------------

conf: # [object] [required]
  type: # [string]
  searchHead: # [string] [required] Search head - Search head base URL. Can be ar
  search: # [string] Search - Examples: 'index=myAppLogs level=error channel=myAp
  earliest: # [string] Earliest - The earliest time boundary for the search. Can
  latest: # [string] Latest - The latest time boundary for the search. Can be an
  endpoint: # [string] [required] Search endpoint - REST API used to create a sea
```

```
outputMode: # [string] [required] Output mode - Format of the returned output
collectRequestParams: # [array] Extra parameters - Optional collect request par
  - name: # [string] Parameter Name
    value: # [string] Value - JavaScript expression to compute the parameter's
collectRequestHeaders: # [array] Extra headers - Optional collect request heade
  - name: # [string] Header Name
    value: # [string] Value - JavaScript expression to compute the header's val
authentication: # [string] [required] Authentication - Authentication method fo

# -------------- if authentication is basic ---------------

username: # [string] Username - Basic authentication username
password: # [string] Password - Basic authentication password

# ---------------------------------------------------------


# -------------- if authentication is basicSecret ---------------

credentialsSecret: # [string] Credentials secret - Select or create a stored se

# ---------------------------------------------------------


# -------------- if authentication is token ---------------

token: # [string] Bearer token

# ---------------------------------------------------------


# -------------- if authentication is tokenSecret ---------------

tokenSecret: # [string] Bearer token secret - Select or create a stored secret

# ---------------------------------------------------------


# -------------- if authentication is login ---------------

loginUrl: # [string] Login URL - URL to use for login API call. This call is ex
username: # [string] Username
password: # [string] Password
```

```
loginBody: # [string] POST body - Template for POST body to send with login rec
tokenRespAttribute: # [string] Token attribute - Path to token attribute in log
authHeaderExpr: # [string] Authorize Expression - JavaScript expression to comp

# -------------------------------------------------------


# -------------- if authentication is loginSecret ---------------

loginUrl: # [string] Login URL - URL to use for login API call, this call is ex
credentialsSecret: # [string] Credentials secret - Select or create a stored se
loginBody: # [string] POST Body - Template for POST body to send with login rec
tokenRespAttribute: # [string] Token Attribute - Path to token attribute in log
authHeaderExpr: # [string] Authorize expression - JavaScript expression to comp

# -------------------------------------------------------

timeout: # [number; maximum: 1800] Request timeout (secs) - HTTP request inacti
useRoundRobinDns: # [boolean] Round-robin DNS - Use round-robin DNS lookup. Sui
disableTimeFilter: # [boolean] Disable time filter - Disable collector event ti
rejectUnauthorized: # [boolean] Reject unauthorized certificates - Reject certi
retryRules: # [object]
  type: # [string] Retry type - Algorithm to use when performing HTTP retries

  # -------------- if type is static ---------------

  interval: # [number; maximum: 20000] Wait (ms) - Time interval between retrie
  limit: # [number; maximum: 20] Retry limit - The maximum number of times to r
  codes: # [array of numbers] Retry HTTP codes - List of HTTP codes that trigge
  enableHeader: # [boolean] Honor Retry-After header - Honor any Retry-After he
  retryConnectTimeout: # [boolean] Retry connection timeout - Make a single ret
  retryConnectReset: # [boolean] Retry connection reset - Retry request when a

  # -------------------------------------------------------


  # -------------- if type is backoff ---------------

  interval: # [number; maximum: 20000] Initial retry interval (ms) - Time inter
  limit: # [number; maximum: 20] Retry limit - The maximum number of times to r
  multiplier: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base for
  codes: # [array of numbers] Retry HTTP codes - List of HTTP codes that trigge
  enableHeader: # [boolean] Honor Retry-After header - Honor any Retry-After he
```

```
      retryConnectTimeout: # [boolean] Retry connection timeout - Make a single ret
      retryConnectReset: # [boolean] Retry connection reset - Retry request when a

      # ------------------------------------------------------


  # ------------------------------------------------------

  destructive: # [boolean] Destructive - Delete any files collected (where applicab
  encoding: # [string] Encoding - Character encoding to use when parsing ingested d
input: # [object]
  type: # [string]
  breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event-br
  staleChannelFlushMs: # [number; minimum: 10, maximum: 43200000] Event Breaker buf
  sendToRoutes: # [boolean] Send to Routes - Send events to normal routing and even

  # -------------- if sendToRoutes is true ---------------

  pipeline: # [string] Pre-Processing Pipeline - An optional Pipeline to process re

  # ------------------------------------------------------


  # -------------- if sendToRoutes is false ---------------

  pipeline: # [string] Pipeline - Pipeline to process results
  output: # [string] Destination - Destination to send results to

  # ------------------------------------------------------

  preprocess: # [object]
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    command: # [string] Command - Command to feed the data through (via stdin) and
    args: # [array of strings] Arguments - Arguments to be added to the custom comm

    # ------------------------------------------------------

  throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to throttl
  metadata: # [array] Fields - Fields to add to events from this input
    - name: # [string] Field Name
```

```
      value: # [string] Value - JavaScript expression to compute field's value, enc
description: # [string] Description
type: # [string] Job type
ttl: # [string] Time to live - Time to keep the job's artifacts on disk after job c
ignoreGroupJobsLimit: # [boolean] Ignore Worker Group job limits - When enabled, th
removeFields: # [array of strings] Remove Discover fields - List of fields to remov
resumeOnBoot: # [boolean] Resume job on boot - Resume the ad hoc job if a failure c
environment: # [string] Environment - Optionally, enable this config only on a spec
schedule: # [object] Schedule - Configuration for a scheduled job
  enabled: # [boolean] Enabled - Enable to configure scheduling for this Collector

  # -------------- if enabled is true ---------------

  cronSchedule: # [string] Cron schedule - A cron schedule on which to run this job
  maxConcurrentRuns: # [number; minimum: 1] Concurrent run limit - The maximum numb
  skippable: # [boolean] Skippable - Skippable jobs can be delayed, up to their nex
  run: # [object] Run settings

    # -------------- if type is collection ---------------

    rescheduleDroppedTasks: # [boolean] Reschedule tasks - Reschedule tasks that fa
    maxTaskReschedule: # [number; minimum: 1] Task reschedule limit - Maximum numbe
    logLevel: # [string] Log level - Level at which to set task logging
    jobTimeout: # [string] Job timeout - Maximum time the job is allowed to run. Ti
    mode: # [string] Mode - Job run mode. Preview will either return up to N matchi

    # -------------- if mode is list ---------------

    discoverToRoutes: # [boolean] Send to Routes - Send discover results to Routes

    # -----------------------------------------------------



    # -------------- if mode is preview ---------------

    capture: # [object] Capture Settings
      duration: # [number] Capture time (sec) - Amount of time to keep capture oper
      maxEvents: # [number; minimum: 1, maximum: 10000] Capture up to N events - Ma
      level: # [string] Where to capture

    # -----------------------------------------------------

    timeRangeType: # [string] Time range
```

```
      # -------------- if timeRangeType is absolute ---------------

      timestampTimezone: # [string] Range timezone - Timezone to use for Earliest and

      # ------------------------------------------------------

      earliest: # [number,string] Earliest - Earliest time to collect data for the se
      latest: # [number,string] Latest - Latest time to collect data for the selected
      timeWarning: # [object]
      expression: # [string] Filter - A filter for tokens in the provided collect pat
      minTaskSize: # [string] Lower task bundle size - Limits the bundle size for sma
        if your lower bundle size is 1MB, you can bundle up to five 200KB files into
      maxTaskSize: # [string] Upper task bundle size - Limits the bundle size for fil
        you can bundle up to five 2MB files into one task. Files greater than this si

    # ------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl Ed
executor_job: # [object]
  executor: # [object]
    type: # [string] Executor type - The type of executor to run
    storeTaskResults: # [boolean] Store task results - Determines whether or not to w
    conf: # [object] Executor-specific settings
  description: # [string] Description
  type: # [string] Job type
  ttl: # [string] Time to live - Time to keep the job's artifacts on disk after job c
  ignoreGroupJobsLimit: # [boolean] Ignore Worker Group job limits - When enabled, th
  removeFields: # [array of strings] Remove Discover fields - List of fields to remov
  resumeOnBoot: # [boolean] Resume job on boot - Resume the ad hoc job if a failure c
  environment: # [string] Environment - Optionally, enable this config only on a spec
  schedule: # [object] Schedule - Configuration for a scheduled job
    enabled: # [boolean] Enabled - Enable to configure scheduling for this Collector

    # -------------- if enabled is true ---------------

    cronSchedule: # [string] Cron schedule - A cron schedule on which to run this job
    maxConcurrentRuns: # [number; minimum: 1] Concurrent run limit - The maximum numb
    skippable: # [boolean] Skippable - Skippable jobs can be delayed, up to their nex
    run: # [object] Run settings

      # -------------- if type is collection ---------------
```

```
    rescheduleDroppedTasks: # [boolean] Reschedule tasks - Reschedule tasks that fa
    maxTaskReschedule: # [number; minimum: 1] Task reschedule limit - Maximum numbe
    logLevel: # [string] Log level - Level at which to set task logging
    jobTimeout: # [string] Job timeout - Maximum time the job is allowed to run. Ti
    mode: # [string] Mode - Job run mode. Preview will either return up to N matchi

    # -------------- if mode is list ---------------

    discoverToRoutes: # [boolean] Send to Routes - Send discover results to Routes

    # ------------------------------------------------------


    # -------------- if mode is preview ---------------

    capture: # [object] Capture Settings
      duration: # [number] Capture time (sec) - Amount of time to keep capture oper
      maxEvents: # [number; minimum: 1, maximum: 10000] Capture up to N events - Ma
      level: # [string] Where to capture

    # ------------------------------------------------------

    timeRangeType: # [string] Time range

    # -------------- if timeRangeType is absolute ---------------

    timestampTimezone: # [string] Range timezone - Timezone to use for Earliest and

    # ------------------------------------------------------

    earliest: # [number,string] Earliest - Earliest time to collect data for the se
    latest: # [number,string] Latest - Latest time to collect data for the selected
    timeWarning: # [object]
    expression: # [string] Filter - A filter for tokens in the provided collect pat
    minTaskSize: # [string] Lower task bundle size - Limits the bundle size for sma
      if your lower bundle size is 1MB, you can bundle up to five 200KB files into
    maxTaskSize: # [string] Upper task bundle size - Limits the bundle size for fil
      you can bundle up to five 2MB files into one task. Files greater than this si

  # ------------------------------------------------------

  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl Ed
scheduledSearch_job: # [object]
```

```
savedQueryId: # [string] ID of the SavedQuery - Identifies which search query to ru
description: # [string] Description
type: # [string] Job type
ttl: # [string] Time to live - Time to keep the job's artifacts on disk after job c
ignoreGroupJobsLimit: # [boolean] Ignore Worker Group job limits - When enabled, th
removeFields: # [array of strings] Remove Discover fields - List of fields to remov
resumeOnBoot: # [boolean] Resume job on boot - Resume the ad hoc job if a failure c
environment: # [string] Environment - Optionally, enable this config only on a spec
schedule: # [object] Schedule - Configuration for a scheduled job
  enabled: # [boolean] Enabled - Enable to configure scheduling for this Collector

  # -------------- if enabled is true ---------------

  cronSchedule: # [string] Cron schedule - A cron schedule on which to run this job
  maxConcurrentRuns: # [number; minimum: 1] Concurrent run limit - The maximum numb
  skippable: # [boolean] Skippable - Skippable jobs can be delayed, up to their nex
  run: # [object] Run settings

    # -------------- if type is collection ---------------

    rescheduleDroppedTasks: # [boolean] Reschedule tasks - Reschedule tasks that fa
    maxTaskReschedule: # [number; minimum: 1] Task reschedule limit - Maximum numbe
    logLevel: # [string] Log level - Level at which to set task logging
    jobTimeout: # [string] Job timeout - Maximum time the job is allowed to run. Ti
    mode: # [string] Mode - Job run mode. Preview will either return up to N matchi

    # -------------- if mode is list ---------------

    discoverToRoutes: # [boolean] Send to Routes - Send discover results to Routes

    # -----------------------------------------------------



    # -------------- if mode is preview ---------------

    capture: # [object] Capture Settings
      duration: # [number] Capture time (sec) - Amount of time to keep capture oper
      maxEvents: # [number; minimum: 1, maximum: 10000] Capture up to N events - Ma
```

17.5.8. Job-Limits.Yml

`job-limits.yml` maintains parameters for Collection jobs and system tasks. In the UI, you can configure these per Fleet at **Fleet Settings** > **General Settings** > **Limits** > **Jobs**.

Where we do not specify or validate a minimum or maximum value, Cribl provides a prudent default. Adjust this initial value to match your architecture and needs.

job-limits.yml

```
# Disable jobs/tasks - When jobs/tasks are disabled, Worker and Edge Nodes on versior
# poll the Leader for tasks. Nodes running 4.4.4 and older will honor the Jobs settir
# are disabled.
# [boolean; default: false]
disableTasks:
# Concurrent job limit - The total number of jobs that may run concurrently
# [number; min: 1; default: 10]
concurrentJobLimit:
# Concurrent system job limit - The total number of system jobs that may run concurre
# [number; min: 1; default: 10]
concurrentSystemJobLimit:
# Concurrent scheduled job limit - The total number of scheduled jobs that can run cc
# relative to concurrent job limit.
# [number; default: -2]
concurrentScheduledJobLimit:
# Concurrent task limit - The total number of tasks that a Worker Process can run cor
# [number; min: 0; default: 2]
concurrentTaskLimit:
# Concurrent system task limit - The number of system tasks that a Worker Process car
# [number; min: 0; default: 1]
concurrentSystemTaskLimit:
# Task usage percentage limit - Value from 0 to 1 representing the percentage of tota
# system a job may consume
# [number; min: 0; max: 1; default: 0.5]
maxTaskPerc:
# Task poll timeout - The number of milliseconds the Worker's task handler will wait
# before retrying the request for a task
# [number; min: 10000; default: 60000]
taskPollTimeoutMs:
# Artifact reaper period - Time period at which the system attempts to reap stale dis
# to the jobs
# [string; default: 30m]
jobArtifactsReaperPeriod:
# Finished job artifacts limit - Maximum number of finished job artifacts to keep on
# [number; min: 0; default: 100]
finishedJobArtifactsLimit:
```

17.5.9. Leader.Yml

`leader.yml` contains configuration for a secondary Leader and is located under `$CRIBL_HOME/local/cribl` in the volume of that Leader.

This configuration file is created when you have configured a Leader for failover and the new Leader becomes active in case of failover.

leader.yml

```
# Host/IP address to bind to - The host address for the instance
# [string; default: 0.0.0.0; required]
host:
# Port number to bind to - The port number for the instance
# [number; default: 4200; required]
port:
# Protocol - Communication protocol
# [string]
protocol:
# Auth Token - Authentication token for secure communication
# [string]
authToken:
# IP allowlist regex - Regex matching IP addresses that are allowed to establish a cc
# [string; default: /.*/]
ipWhitelistRegex:
# Proxy settings - SOCKS proxy configuration
proxy:
# Compression - Data compression method
# [string]
compression:
# Connection timeout - Timeout for establishing connections
# [number]
connectionTimeout:
# Write timeout - Timeout for write operations
# [number]
writeTimeout:
# Max buffer bytes - Maximum buffer size in bytes
# [number]
maxBufferBytes:
# Forward to Leader API - Controls whether distributed traffic gets proxied to Leader
# [boolean; default: true]
forwardToLeaderApi:
# TLS configuration - TLS/SSL settings
tls:
  # Disabled - Whether TLS is disabled
  # [boolean]
  disabled:
  # Certificate Name - Name of the certificate to use
  # [string]
  certificateName:
  # Private Key Path - Path to private key file
  # [string]
```

```
privKeyPath:
# Passphrase - Passphrase for private key
# [string]
passphrase:
# Certificate Path - Path to certificate file
# [string]
certPath:
# CA Path - Path to certificate authority file
# [string]
caPath:
# Request Certificate - Whether to request client certificates
# [boolean]
requestCert:
# Reject Unauthorized - Whether to reject unauthorized certificates
```

17.5.10. LIMITS.YML

`limits.yml` maintains parameters for the system. In a distributed deployment, these parameters are configured on both the Leader and the Workers.

In the UI, you can configure them:

- On the Leader at **Settings** > **Global** > **General Settings > Limits**.
- On a single instance at **Settings** > **General Settings > Limits**.
- On Fleets at `<Group/Fleet-name>` > **Fleet Settings** > **General Settings > Limits**.

Where we do not specify or validate a minimum or maximum value, Cribl provides a prudent default. Adjust this initial value to match your architecture and needs.

limits.yml

```
# Default managed node heartbeat period - How many seconds a managed node will wait t
# the Cribl control plane
# [number; min: 1]
defaultHeartbeatSeconds:
# Samples
samples:
  # Sample size limit - Maximum file size for the sample, in binary units (KB, MB). (
  # [string]
  maxSize:
# Min free disk space - The minimum amount of disk space in the system before various
# measures to prevent disk usage (KB, MB, etc.)
# [string; default: 5 GB]
minFreeSpace:
# Worker Process PQ size limit - Highest accepted value for individual integrations'
# queue size' parameters. Consult Cribl Support before increasing the default 1 TB.
# [string; default: 1TB]
maxPQSize:
# Metrics GC period - The interval at which the system attempts to free memory by pru
# the Stream system metrics store
# [string; default: 60s]
metricsGCPeriod:
# Number of metrics allowed - The system's total allowed number of metric series. Cor
# before updating this value.
# [number; default: 1000000]
maxMetrics:
# Worker metrics limit - The maximum number of metric series allowed per worker. Cons
# before updating this value.
# [number; default: 100000]
workerMaxMetrics:
# Metrics cardinality limit - The system's allowed number of permutations of a given
# [number; default: 1000]
metricsMaxCardinality:
# Metrics disk space limit - Maximum allowed disk space for persisting metrics to dis
# [string; default: 64GB]
metricsMaxDiskSpace:
# Metrics worker tracking - List of metric names for which to disable tracking of Wor
# Supports wildcards.
# [array; default: host.*, source.*, sourcetype.*, index.*]
metricsWorkerIdBlacklist:
# Metrics never-drop list - List metric names for which to ensure delivery. Supports
# [array; default: total.*]
metricsNeverDropList:
```

```
# Disable field metrics - List of event fields for which to disable metric collectior
# [array; default: source, host, index, sourcetype]
metricsFieldsBlacklist:
# Metrics directory - Directory to store metrics on disk
# [string; default: $CRIBL_STATE_DIR/metrics]
metricsDirectory:
# CPU profile TTL - The time to live for collected CPU profiles
# [string; default: 30m]
cpuProfileTTL:
# Event metadata sources - List of event metadata sources to enable
eventsMetadataSources:
# Metrics to send from Edge Nodes - Choose a set of metrics for Edge Nodes to send to
# a custom set. Details about pre-defined sets are in [the docs](https://docs.cribl.i
# One of: basic | detailed | none | custom
# [string; default: basic]
edgeMetricsMode:
# Metrics expression - JavaScript expression to filter metrics
# [string; required]
edgeMetricsCustomExpression:
# Persist metrics - Disable to stop writing metrics to disk
# [boolean; default: true]
enableMetricsPersistence:
# Edge Nodes estimated count - Used to pre-allocate connection handlers. Changing thi
# currently running connection handlers, causing all Edge Nodes to reconnect. A blank
# default, which starts a handler on demand when an Edge Node attempts to connect.
# [number; min: 0; max: 250000; default: 0]
edgeNodesCount:
# Disable Node persistence - Do not persist connection data across restarts. Caution:
# [boolean; default: false]
enableWorkerInMemory:
# Enable KVStore WAL - Enable the write-ahead log for the KVStore
# [boolean; default: false]
kvStoreEnableWAL:
# KVStore persist period - How often the KVStore syncs itself to disk. When WAL mode
# the KVStore are synced to disk immediately as they happen; however, the WAL is merg
# KVStore on this interval.
# [number; min: 10000]
kvStoreServicePeriod:
# KVStore WAL file bytes limit - The amount of bytes to allow a WAL file to get to be
# back into the KVStore. This may result in performance degradation if adjusted.
# [number; min: 1024; default: 5242880]
kvStoreWALMaxBytes:
# Config bundle download timeout (secs) - The maximum time a Worker will wait for a s
```

```
# connection before canceling a download of a new configuration bundle. A 0 value mea
# which could cause Workers to hang.
# [number; min: 0; default: 0]
bundleDownloadTimeoutSeconds:
# Minimum reconnect interval - The minimum time to wait before attempting to reconnec
# being disconnected. The interval is doubled with each failed reconnect attempt unti
# reached.
# [string; default: 2s]
minReconnectInterval:
# Maximum reconnect interval - The maximum time to wait before attempting to reconnec
# being disconnected.
# [string; default: 1m]
maxReconnectInterval:
# Random reconnect interval - Upper limit on the random time added to each reconnect
# non-zero interval to randomly spread out reconnection attempts. Useful for Leaders
# Worker or Edge nodes.
# [string; default: 0s]
randomReconnectInterval:
# Consult Cribl Support before adjusting this advanced setting. The maximum number of
# a worker node is allowed to ingest. If 0 or not configured, no rate limiting is app
# means automatic determination based on configurations.
# [string; default: '']
rateLimitBytesPerSecond:
# Use Windows tools to collect process info - Enable PowerShell to collect process in
# using the native API
# [boolean; default: false]
enableProcessInfoUsingWindowsTools:
# Lookup file max size - Maximum allowed disk space for persisting single lookup data
# [string; default: 500MB]
lookupMaxSize:
# Lookup files disk limit - Maximum allowed disk space for persisting lookup data and
# [string; default: 2GB]
lookupMaxTotalSize:
```

17.5.11. Logger.Yml

`logger.yml` maintains logging levels and redactions, per channel. In the UI, you can configure these at **Settings** > **Global** > **System** > **Logging**.

logger.yml

```
# Logger Configuration ID
# [string]
id:
# Default redact fields - List of default field names to redact in internal logs
defaultRedactFields:
# Additional redact fields - List of additional field names to redact in internal log
redactFields:
# Custom redact string - Literal redaction string to apply on selected field values.
# [string]
redactLabel:
# Log channels - Configuration for individual log channels
channels:
# TTL configuration - Time-to-live configuration for log levels
ttlConfig:
  # Log level - Level at which to log messages
  # One of: error | warn | info | http | verbose | debug | silly
  # [string; required]
  level:
  # Time to live - Duration in seconds after which log level will revert to default
  # [number; required]
  ttl:
# Max log file size - Maximum log file size in bytes
# [number]
maxSizeBytes:
# Rate limit - Rate limit for log messages (0 to disable)
# [number]
limitRate:
```

Copy

17.5.12. Mappings.Yml And Fleet-Mappings.Yml

Mapping ruleset configurations for Stream Workers and Edge Node are stored in `$CRIBL_HOME/local/cribl/mappings.yml` and `$CRIBL_HOME/local/cribl/fleet-mappings.yml`, respectively.

mappings.yml or fleet-mappings.yml

```
# Mapping ID - Unique identifier for the mapping ruleset
# [string; required]
id:
# Configuration - Mapping ruleset configuration
conf:
  # Functions - List of functions to pass data through
  functions:
# Active - Whether this mapping ruleset is active
# [boolean]
active:
```

Copy

17.5.13. Messages.Yml

`messages.yml` stores messages displayed in the UI's `Messages` fly-out.

logger.yml

```
# Message ID - Unique identifier for the message
# [string]
id:
# Message Type - Type of the bulletin message
# [string]
type:
# Message Text - Content of the message
# [string]
message:
# Severity - Severity level of the message
# [string]
severity:
# Created - Timestamp when the message was created
# [number]
created:
# TTL - Time to live for the message
# [number]
ttl:
```

Copy

17.5.14. Notifications.Yml

`notifications.yml` contains settings for Notifications.

notifications.yml

```yaml
# Notification Targets - Configuration for notification targets
targets:
  # Output ID - Unique identifier for this notification target
  # [string; required]
  id:
  # Output Type - Type of notification target
  # [string; required]
  type:
  # System fields - Fields to automatically add to events, such as cribl_pipe. Suppor
  # [array; default: cribl_host]
  systemFields:
  # Webhook configuration (webhook type only)
  # URL - Webhook endpoint URL
  # [string; required for webhook]
  url:
  # Format - Data format for webhook payload
  # [string; default: ndjson for webhook]
  format:
  # Method - HTTP method for webhook requests
  # [string; default: POST for webhook]
  method:
  # Authentication type - Authentication method for webhook requests
  # [string; default: none for webhook]
  authType:
  # Token - Authentication token (when authType is token)
  # [string]
  token:
  # Username - Username for basic authentication (when authType is basic)
  # [string]
  username:
  # Password - Password for basic authentication (when authType is basic)
  # [string]
  password:
  # Source expression - Expression to evaluate on events to generate output notificat
  # [string]
  srcExpression:
  # Slack configuration (slack type only)
  # Slack webhook URL - Slack incoming webhook URL
  # [string; required for slack]
  slackUrl:
  # PagerDuty configuration (pager_duty type only)
  # Routing key - PagerDuty integration key
```

```
# [string; required for pager_duty]
routingKey:
# Severity - Alert severity level
# [string; required for pager_duty]
severity:
# Group - Alert group identifier
# [string]
group:
# Class - Alert class identifier
# [string]
class:
# Component - Alert component identifier
# [string]
component:
# SNS configuration (sns type only)
# Topic ARN - AWS SNS topic ARN
# [string; required for sns]
topicArn:
# Region - AWS region
# [string; required for sns]
region:
# AWS authentication method - Method for AWS authentication
# [string; default: auto for sns]
awsAuthenticationMethod:
# Destination type - Type of SNS destination
# [string; required for sns]
destinationType:
# Topic type - SNS topic type
# [string; default: standard for sns]
topicType:
# Message group ID - Message group ID for FIFO topics
# [string; required for sns]
messageGroupId:
# Allow list - List of allowed phone numbers or email addresses
allowList:
# Assume role ARN - ARN of the role to assume
# [string]
assumeRoleArn:
# Enable assume role - Whether to enable role assumption
# [boolean]
enableAssumeRole:
# SMTP configuration (smtp type only)
# Host - SMTP server hostname
```

```
  # [string; required for smtp]
  host:
  # Port - SMTP server port
  # [number; required for smtp]
  port:
  # SMTP username - Username for SMTP authentication
  # [string; required for smtp]
  smtpUsername:
  # SMTP password - Password for SMTP authentication
  # [string; required for smtp]
  smtpPassword:
  # From - Sender email address
  # [string; required for smtp]
  from:
  # TLS - TLS configuration for SMTP
  tls:
    # Reject unauthorized - Whether to reject unauthorized certificates
    # [boolean; default: true]
    rejectUnauthorized:
    # Minimum TLS version - Minimum TLS version
    # [string]
    minVersion:
    # Maximum TLS version - Maximum TLS version
    # [string]
    maxVersion:
  # Router configuration (router type only)
  # Routes - Routing rules for notifications
  routes:
  # Bulletin message configuration (bulletin_message type only)
  # Message - Bulletin message content
  # [string]
  message:
  # Notifications log configuration (notifications_log type only)
  # Log notifications to internal log
  logNotifications:
# Notifications - Individual notification configurations
notifications:
  # Notification ID - Unique identifier for this notification
  # [string; required]
  id:
  # Disabled - Whether this notification is disabled
  # [boolean; default: false]
  disabled:
```

```
# Condıtıon - When to trıgger thıs notıfıcatıon
# [string; required]
condition:
# Notification targets - Targets to send any Notifications to
targets:
# Target configuration - Configuration for specific targets
```

17.5.15. Outpost.Yml

`outpost.yml` contains configuration for a Cribl Outpost instance.

outpost.yml

```
# Settings for the internal distributed-management interface
listener:
  # Host - Host/IP address to bind to
  # [string; required]
  host:
  # Port - Port number to bind to
  # [number; required]
  port:
```

17.5.16. Outputs.Yml

`outputs.yml` contains configuration settings for Cribl Edge Destinations.

outputs.yml

```
outputs: # [object]
  default_output: # [object]
    type: # [string] Output Type
    defaultId: # [string,null] Default Output ID - ID of the default output. This wil
    pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
    systemFields: # [array of strings] System fields - Fields to automatically add to
    environment: # [string] Environment - Optionally, enable this config only on a sp
    streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
  webhook_output: # [object]
    type: # [string] Output Type
    method: # [string] Method - The method to use when sending events
    format: # [string] Format - How to format events before sending out

    # -------------- if format is custom ---------------

    customSourceExpression: # [string] Source expression - Expression to evaluate on
    customDropWhenNull: # [boolean] Drop when null - Whether to drop events when the
    customEventDelimiter: # [string] Event delimiter - Delimiter string to insert bet
    customContentType: # [string] Content type - Content type to use for request. Def
    customPayloadExpression: # [string] Batch expression - Expression specifying how

    # ---------------------------------------------------------


    # -------------- if format is advanced ---------------

    advancedContentType: # [string] Content type - HTTP content-type header value
    formatEventCode: # [string] Format inbound event - Custom JavaScript code to form
    formatPayloadCode: # [string] Format outbound payload - Optional JavaScript code

    # ---------------------------------------------------------


    keepAlive: # [boolean] Keep alive - Disable to close the connection immediately a
    concurrency: # [number; minimum: 1, maximum: 32] Request concurrency - Maximum nu
    maxPayloadSizeKB: # [number; minimum: 1024, maximum: 512000] Body size limit (KB)
    maxPayloadEvents: # [number] Events-per-request limit - Maximum number of events
    compress: # [boolean] Compress - Compress the payload body before sending
    rejectUnauthorized: # [boolean] Validate server certs - Reject certificates not a
        Enabled by default. When this setting is also present in TLS Settings (Client
        that value will take precedence.
    timeoutSec: # [number; minimum: 1, maximum: 9007199254740991] Request timeout - A
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
```

```
extraHttpHeaders: # [array] Extra HTTP headers - Headers to add to all events. Yo
  - name: # [string] Field Name
    value: # [string] Field Value
useRoundRobinDns: # [boolean] Round-robin DNS - Enable round-robin DNS lookup. Wh
failedRequestLoggingMode: # [string] Failed request logging mode - Data to log wh
safeHeaders: # [array of strings] Safe headers - List of headers that are safe to
responseRetrySettings: # [array] Settings for failed HTTP requests - Automatically
  - httpStatus: # [number; minimum: 100, maximum: 599] HTTP status code - The HTT
    initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How l
    backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base fo
    maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) -
timeoutRetrySettings: # [object]
  timeoutRetry: # [boolean] Retry timed-out HTTP requests

  # -------------- if timeoutRetry is true ---------------

  initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How lon
  backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base for
  maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) - Th

  # -------------------------------------------------------

responseHonorRetryAfterHeader: # [boolean] Honor Retry-After header - Honor any R
onBackpressure: # [string] Backpressure behavior - How to handle events when all

# -------------- if onBackpressure is queue ---------------

pqMaxFileSize: # [string] File size limit - The maximum size to store in each que
pqMaxSize: # [string] Queue size limit - The maximum disk space that the queue ca
pqPath: # [string] Queue file path - The location for the persistent queue files.
pqCompress: # [string] Compression - Codec to use to compress the persisted data
pqOnBackpressure: # [string] Queue-full behavior - How to handle events when the
pqMode: # [string] Mode - In Error mode, PQ writes events to the filesystem if th

# -------------------------------------------------------

authType: # [string] Authentication type - Authentication method to use for the H

# -------------- if authType is basic ---------------

username: # [string] Username
password: # [string] Password
```

```
# ------------------------------------------------------

# -------------- if authType is token ---------------

token: # [string] Token - Bearer token to include in the authorization header

# ------------------------------------------------------

# -------------- if authType is credentialsSecret ---------------

credentialsSecret: # [string] Credentials secret - Select or create a secret that

# ------------------------------------------------------

# -------------- if authType is textSecret ---------------

textSecret: # [string] Token (text secret) - Select or create a stored text secre

# ------------------------------------------------------

# -------------- if authType is oauth ---------------

loginUrl: # [string] Login URL - URL for OAuth
secretParamName: # [string] OAuth Secret parameter name - Secret parameter name t
secret: # [string] OAuth secret - Secret parameter value to pass in request body
tokenAttributeName: # [string] Token attribute name - Name of the auth token attr
spacer: # [null]
authHeaderExpr: # [string] Authorize expression - JavaScript expression to comput
tokenTimeoutSecs: # [number; minimum: 1, maximum: 300000] Refresh interval (secs.
oauthParams: # [array] OAuth parameters - Additional parameters to send in the OA
  - name: # [string] Name - OAuth parameter name
    value: # [string] Value - OAuth parameter value
oauthHeaders: # [array] OAuth headers - Additional headers to send in the OAuth l
  - name: # [string] Name - OAuth header name
    value: # [string] Value - OAuth header value

# ------------------------------------------------------

tls: # [object] TLS settings (client side)
```

```
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    servername: # [string] Server name (SNI) - Server name for the SNI (Server Name
    certificateName: # [string] Certificate - The name of the predefined certificat
    caPath: # [string] CA certificate path - Path on client in which to find CA cer
    privKeyPath: # [string] Private key path (mutual auth) - Path on client in whic
    certPath: # [string] Certificate path (mutual auth) - Path on client in which t
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
    minVersion: # [string] Minimum TLS version
    maxVersion: # [string] Maximum TLS version

    # ---------------------------------------------------------

  totalMemoryLimitKB: # [number] Buffer memory limit (KB) - Maximum total size of t
  loadBalanced: # [boolean] Load balancing - Enable for optimal performance. Even i

  # -------------- if loadBalanced is false ---------------

  url: # [string] Webhook URL - URL of a webhook endpoint to send events to, such a
  useRoundRobinDns: # [boolean] Round-robin DNS - Enable round-robin DNS lookup. Wh

  # ---------------------------------------------------------


  # -------------- if loadBalanced is true ---------------

  excludeSelf: # [boolean] Exclude current host IPs - Exclude all IPs of the curren
  urls: # [array] Webhook URLs
    - url: # [string] Webhook LB URL - URL of a webhook endpoint to send events to,
      weight: # [number] Load Weight - Assign a weight (>0) to each endpoint to inc
  dnsResolvePeriodSec: # [number; maximum: 86400] DNS resolution period (seconds) -
  loadBalanceStatsPeriodSec: # [number; minimum: 10] Load balance stats period (sec

  # ---------------------------------------------------------

  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
sentinel_output: # [object]
```

```
keepAlive: # [boolean] Keep alive - Disable to close the connection immediately a
concurrency: # [number; minimum: 1, maximum: 32] Request concurrency - Maximum nu
maxPayloadSizeKB: # [number; minimum: 100, maximum: 1000] Body size limit (KB) -
maxPayloadEvents: # [number] Events-per-request limit - Maximum number of events
compress: # [boolean] Compress - Compress the payload body before sending
rejectUnauthorized: # [boolean] Validate server certs - Reject certificates not a
    Enabled by default. When this setting is also present in TLS Settings (Client
    that value will take precedence.
timeoutSec: # [number; minimum: 1, maximum: 9007199254740991] Request timeout - A
flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
extraHttpHeaders: # [array] Extra HTTP headers - Headers to add to all events. Yo
  - name: # [string] Field Name
    value: # [string] Field Value
useRoundRobinDns: # [boolean] Round-robin DNS - Enable round-robin DNS lookup. Wh
failedRequestLoggingMode: # [string] Failed request logging mode - Data to log wh
safeHeaders: # [array of strings] Safe headers - List of headers that are safe to
responseRetrySettings: # [array] Settings for failed HTTP requests - Automaticall
  - httpStatus: # [number; minimum: 100, maximum: 599] HTTP status code - The HTT
    initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How l
    backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base fo
    maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) -
timeoutRetrySettings: # [object]
  timeoutRetry: # [boolean] Retry timed-out HTTP requests

  # -------------- if timeoutRetry is true ---------------

  initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How lon
  backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base for
  maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) - Th

  # -------------------------------------------------------

responseHonorRetryAfterHeader: # [boolean] Honor Retry-After header - Honor any R
onBackpressure: # [string] Backpressure behavior - How to handle events when all

# -------------- if onBackpressure is queue ---------------

pqMaxFileSize: # [string] File size limit - The maximum size to store in each que
pqMaxSize: # [string] Queue size limit - The maximum disk space that the queue ca
pqPath: # [string] Queue file path - The location for the persistent queue files.
pqCompress: # [string] Compression - Codec to use to compress the persisted data
pqOnBackpressure: # [string] Queue-full behavior - How to handle events when the
pqMode: # [string] Mode - In Error mode, PQ writes events to the filesystem if th
```

```
  # ------------------------------------------------------

  loginUrl: # [string] [required] Login URL - URL for OAuth
  secret: # [string] [required] OAuth secret - Secret parameter value to pass in re
  spacer: # [null]
  client_id: # [string] [required] Client ID - JavaScript expression to compute the
  scope: # [string] Scope - Scope to pass in the OAuth request
  type: # [string] Output Type
  endpointURLConfiguration: # [string] Endpoint configuration - Enter the data coll

  # -------------- if endpointURLConfiguration is url ---------------

  url: # [string] URL - URL to send events to. Can be overwritten by an event's __u

  # ------------------------------------------------------


  # -------------- if endpointURLConfiguration is ID ---------------

  dcrID: # [string] Data collection rule ID - Immutable ID for the Data Collection
  dceEndpoint: # [string] Data collection endpoint - Data collection endpoint (DCE)
  streamName: # [string] Stream name - The name of the stream (Sentinel table) in w

  # ------------------------------------------------------

  totalMemoryLimitKB: # [number] Buffer memory limit (KB) - Maximum total size of t
  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
devnull_output: # [object]
  type: # [string] Output Type
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
syslog_output: # [object]
  type: # [string] Output Type
  protocol: # [string] Protocol - The network protocol to use for sending out syslo

  # -------------- if protocol is tcp ---------------
```

```
loadBalanced: # [boolean] Load balancing - For optimal performance, enable load b
connectionTimeout: # [number] Connection timeout - Amount of time (milliseconds)
writeTimeout: # [number] Write timeout - Amount of time (milliseconds) to wait fo
tls: # [object] TLS settings (client side)
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  rejectUnauthorized: # [boolean] Validate server certs - Reject certificates tha
                trusted CA (such as the system's). Defaults to Enabled. Overrides
  servername: # [string] Server name (SNI) - Server name for the SNI (Server Name
  certificateName: # [string] Certificate - The name of the predefined certificat
  caPath: # [string] CA certificate path - Path on client in which to find CA cer
  privKeyPath: # [string] Private key path (mutual auth) - Path on client in whic
  certPath: # [string] Certificate path (mutual auth) - Path on client in which t
  passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
  minVersion: # [string] Minimum TLS version
  maxVersion: # [string] Maximum TLS version

  # ----------------------------------------------------------

onBackpressure: # [string] Backpressure behavior - How to handle events when all

# ----------------------------------------------------------



# -------------- if protocol is udp ---------------

host: # [string] Address - The hostname of the receiver
port: # [number; maximum: 65535] Port - The port to connect to on the provided ho
maxRecordSize: # [number; minimum: 1, maximum: 65535] Record size limit - Maximum
udpDnsResolvePeriodSec: # [number; maximum: 86400] DNS resolution period (sec) -

# ----------------------------------------------------------

facility: # [integer] Facility - Default value for message facility. Will be over
severity: # [integer] Severity - Default value for message severity. Will be over
appName: # [string] App name - Default name for device or application that origin
messageFormat: # [string] Message format - The syslog message format depending on
timestampFormat: # [string] Timestamp format - Timestamp format to use when seria
throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to throttl
octetCountFraming: # [boolean] Octet count framing - Prefix messages with the byt
```

```
  logFailedRequests: # [boolean] Log failed requests to disk - Use to troubleshoot
  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
splunk_output: # [object]
  type: # [string] Output Type
  host: # [string] Address - The hostname of the receiver
  port: # [number; maximum: 65535] [required] Port - The port to connect to on the
  nestedFields: # [string] Nested field serialization - How to serialize nested fie
  throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to throttl
  connectionTimeout: # [number] Connection timeout - Amount of time (milliseconds)
  writeTimeout: # [number] Write timeout - Amount of time (milliseconds) to wait fo
  tls: # [object] TLS settings (client side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    rejectUnauthorized: # [boolean] Validate server certs - Reject certificates tha
                  trusted CA (such as the system's). Defaults to Enabled. Overrides
    servername: # [string] Server name (SNI) - Server name for the SNI (Server Name
    certificateName: # [string] Certificate - The name of the predefined certificat
    caPath: # [string] CA certificate path - Path on client in which to find CA cer
    privKeyPath: # [string] Private key path (mutual auth) - Path on client in whic
    certPath: # [string] Certificate path (mutual auth) - Path on client in which t
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
    minVersion: # [string] Minimum TLS version
    maxVersion: # [string] Maximum TLS version

    # ------------------------------------------------------

  enableMultiMetrics: # [boolean] Output multiple metrics - Output metrics in multi
  enableACK: # [boolean] Minimize in-flight data loss - Check if indexer is shuttin

  # -------------- if enableACK is false ---------------

  maxFailedHealthChecks: # [number] Failed health check limit - Maximum number of t

  # ------------------------------------------------------

  logFailedRequests: # [boolean] Log failed requests to disk - Use to troubleshoot
  maxS2Sversion: # [string] Max S2S version - The highest S2S protocol version to a
```

```
  # -------------- if maxS2Sversion is v4 ---------------

  compress: # [string] Compression - Controls whether the sender should send compre

  # ------------------------------------------------------

  onBackpressure: # [string] Backpressure behavior - How to handle events when all

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] File size limit - The maximum size to store in each que
  pqMaxSize: # [string] Queue size limit - The maximum disk space that the queue ca
  pqPath: # [string] Queue file path - The location for the persistent queue files.
  pqCompress: # [string] Compression - Codec to use to compress the persisted data
  pqOnBackpressure: # [string] Queue-full behavior - How to handle events when the
  pqMode: # [string] Mode - In Error mode, PQ writes events to the filesystem if th

  # ------------------------------------------------------

  authType: # [string] Authentication method - Select Manual to enter an auth token
  authToken: # [string] Auth token - Shared secret token to use when establishing a

  # -------------- if authType is manual ---------------


  # ------------------------------------------------------

  textSecret: # [string] Auth token (text secret) - Select or create a stored text

  # -------------- if authType is secret ---------------


  # ------------------------------------------------------

  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
splunk_lb_output: # [object]
  type: # [string] Output Type
  dnsResolvePeriodSec: # [number; maximum: 86400] DNS resolution period (seconds) -
```

```
loadBalanceStatsPeriodSec: # [number; minimum: 10] Load balance stats period (sec
maxConcurrentSenders: # [number] Connection limit - Maximum number of concurrent
nestedFields: # [string] Nested field serialization - How to serialize nested fie
throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to throttl
connectionTimeout: # [number] Connection timeout - Amount of time (milliseconds)
writeTimeout: # [number] Write timeout - Amount of time (milliseconds) to wait fo
tls: # [object] TLS settings (client side)
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  rejectUnauthorized: # [boolean] Validate server certs - Reject certificates tha
                trusted CA (such as the system's). Defaults to Enabled. Overrides
  servername: # [string] Server name (SNI) - Server name for the SNI (Server Name
  certificateName: # [string] Certificate - The name of the predefined certificat
  caPath: # [string] CA certificate path - Path on client in which to find CA cer
  privKeyPath: # [string] Private key path (mutual auth) - Path on client in whic
  certPath: # [string] Certificate path (mutual auth) - Path on client in which t
  passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
  minVersion: # [string] Minimum TLS version
  maxVersion: # [string] Maximum TLS version

  # ---------------------------------------------------------

enableMultiMetrics: # [boolean] Output multiple metrics - Output metrics in multi
enableACK: # [boolean] Minimize in-flight data loss - Check if indexer is shuttin

# -------------- if enableACK is false ---------------

maxFailedHealthChecks: # [number] Failed health check limit - Maximum number of t

# ---------------------------------------------------------

logFailedRequests: # [boolean] Log failed requests to disk - Use to troubleshoot
maxS2Sversion: # [string] Max S2S version - The highest S2S protocol version to a

# -------------- if maxS2Sversion is v4 ---------------

compress: # [string] Compression - Controls whether the sender should send compre

# ---------------------------------------------------------

onBackpressure: # [string] Backpressure behavior - How to handle events when all
```

```
# -------------- if onBackpressure is queue ---------------

pqMaxFileSize: # [string] File size limit - The maximum size to store in each que
pqMaxSize: # [string] Queue size limit - The maximum disk space that the queue ca
pqPath: # [string] Queue file path - The location for the persistent queue files.
pqCompress: # [string] Compression - Codec to use to compress the persisted data
pqOnBackpressure: # [string] Queue-full behavior - How to handle events when the
pqMode: # [string] Mode - In Error mode, PQ writes events to the filesystem if th

# ------------------------------------------------------

indexerDiscovery: # [boolean] Indexer Discovery - Automatically discover indexers

# -------------- if indexerDiscovery is true ---------------

indexerDiscoveryConfigs: # [object]  - List of configurations to set up indexer d
  site: # [string] [required] Site - Clustering site of the indexers from where i
  masterUri: # [string] Cluster manager URI - Full URI of Splunk cluster manager
  refreshIntervalSec: # [number; minimum: 60, maximum: 86400] [required] Refresh
  rejectUnauthorized: # [boolean] Validate cluster manager certificates - During
  authTokens: # [array] Authentication tokens - Tokens required to authenticate t
    - authType: # [string] Authentication method - Select Manual to enter an auth
  authType: # [string] Authentication method - Select Manual to enter an auth tok
  authToken: # [string] Auth token - Shared secret to be provided by any client (

  # -------------- if authType is manual ---------------


  # ------------------------------------------------------

  textSecret: # [string] Auth token (text secret) - Select or create a stored tex

  # -------------- if authType is secret ---------------


  # ------------------------------------------------------


# ------------------------------------------------------


# -------------- if indexerDiscovery is false ---------------
```

```
  excludeSelf: # [boolean] Exclude current host IPs - Exclude all IPs of the currer
  hosts: # [array] Destinations - Set of Splunk indexers to load-balance data to.
    - host: # [string] Address - The hostname of the receiver
      port: # [number; maximum: 65535] Port - The port to connect to on the provide
      tls: # [string] TLS - Whether to inherit TLS configs from group setting or di
      servername: # [string] TLS Servername - Servername to use if establishing a T
      weight: # [number] Load Weight - Assign a weight (>0) to each endpoint to inc
  dnsResolvePeriodSec: # [number; maximum: 86400] DNS resolution period (seconds) -
  loadBalanceStatsPeriodSec: # [number; minimum: 10] Load balance stats period (sec
  maxConcurrentSenders: # [number] Connection limit - Maximum number of concurrent

  # ------------------------------------------------------

  senderUnhealthyTimeAllowance: # [number; maximum: 60000] Endpoint health fluctuat
  authType: # [string] Authentication method - Select Manual to enter an auth token
  authToken: # [string] Auth token - Shared secret token to use when establishing a

  # -------------- if authType is manual ---------------


  # ------------------------------------------------------

  textSecret: # [string] Auth token (text secret) - Select or create a stored text

  # -------------- if authType is secret ---------------


  # ------------------------------------------------------

  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
splunk_hec_output: # [object]
  type: # [string] Output Type
  loadBalanced: # [boolean] Load balancing - Enable for optimal performance. Even i

  # -------------- if loadBalanced is false ---------------

  url: # [string] Splunk HEC Endpoint - URL to a Splunk HEC endpoint to send events
  useRoundRobinDns: # [boolean] Round-robin DNS - Enable round-robin DNS lookup. Wh
```

```
# -------------------------------------------------------


# -------------- if loadBalanced is true ---------------

excludeSelf: # [boolean] Exclude current host IPs - Exclude all IPs of the currer
urls: # [array] Splunk HEC Endpoints
  - url: # [string] HEC Endpoint - URL to a Splunk HEC endpoint to send events to
    weight: # [number] Load Weight - Assign a weight (>0) to each endpoint to inc
dnsResolvePeriodSec: # [number; maximum: 86400] DNS resolution period (seconds) -
loadBalanceStatsPeriodSec: # [number; minimum: 10] Load balance stats period (sec

# -------------------------------------------------------

nextQueue: # [string] Next Processing Queue - In the Splunk app, define which Spl
tcpRouting: # [string] Default _TCP_ROUTING - In the Splunk app, set the value of
concurrency: # [number; minimum: 1, maximum: 32] Request concurrency - Maximum nu
maxPayloadSizeKB: # [number; minimum: 1024, maximum: 2097152] Body size limit (KB
throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to throttl
maxPayloadEvents: # [number] Events-per-request limit - Maximum number of events
compress: # [boolean] Compress - Compress the payload body before sending
rejectUnauthorized: # [boolean] Validate server certs - Reject certificates not a
    Enabled by default. When this setting is also present in TLS Settings (Client
    that value will take precedence.
timeoutSec: # [number; minimum: 1, maximum: 9007199254740991] Request timeout - A
flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
extraHttpHeaders: # [array] Extra HTTP headers - Headers to add to all events
  - name: # [string] Field Name
    value: # [string] Field Value
failedRequestLoggingMode: # [string] Failed request logging mode - Data to log wh
safeHeaders: # [array of strings] Safe headers - List of headers that are safe to
enableMultiMetrics: # [boolean] Output multi-metrics - Output metrics in multiple
authType: # [string] Authentication method - Select Manual to enter an auth token

# -------------- if authType is manual ---------------

token: # [string] HEC Auth token - Splunk HEC authentication token

# -------------------------------------------------------


# -------------- if authType is secret ---------------
```

```
  textSecret: # [string] HEC Auth token (text secret) - Select or create a stored t

  # ------------------------------------------------------

  responseRetrySettings: # [array] Settings for failed HTTP requests - Automaticall
    - httpStatus: # [number; minimum: 100, maximum: 599] HTTP status code - The HTT
      initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How l
      backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base fo
      maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) -
  timeoutRetrySettings: # [object]
    timeoutRetry: # [boolean] Retry timed-out HTTP requests

    # -------------- if timeoutRetry is true ---------------

    initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How lon
    backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base for
    maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) - Th

    # ------------------------------------------------------

  responseHonorRetryAfterHeader: # [boolean] Honor Retry-After header - Honor any R
  onBackpressure: # [string] Backpressure behavior - How to handle events when all

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] File size limit - The maximum size to store in each que
  pqMaxSize: # [string] Queue size limit - The maximum disk space that the queue ca
  pqPath: # [string] Queue file path - The location for the persistent queue files.
  pqCompress: # [string] Compression - Codec to use to compress the persisted data
  pqOnBackpressure: # [string] Queue-full behavior - How to handle events when the
  pqMode: # [string] Mode - In Error mode, PQ writes events to the filesystem if th

  # ------------------------------------------------------

  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
tcpjson_output: # [object]
  type: # [string] Output Type
  loadBalanced: # [boolean] Load balancing - Use load-balanced destinations
```

```
# -------------- if loadBalanced is false ---------------

host: # [string] Address - The hostname of the receiver
port: # [number; maximum: 65535] Port - The port to connect to on the provided ho

# -------------------------------------------------------


# -------------- if loadBalanced is true ---------------

excludeSelf: # [boolean] Exclude current host IPs - Exclude all IPs of the currer
hosts: # [array] Destinations - Set of hosts to load-balance data to
  - host: # [string] Address - The hostname of the receiver
    port: # [number; maximum: 65535] Port - The port to connect to on the provide
    tls: # [string] TLS - Whether to inherit TLS configs from group setting or di
    servername: # [string] TLS Servername - Servername to use if establishing a T
    weight: # [number] Load Weight - Assign a weight (>0) to each endpoint to inc
dnsResolvePeriodSec: # [number; maximum: 86400] DNS resolution period (seconds) -
loadBalanceStatsPeriodSec: # [number; minimum: 10] Load balance stats period (sec
maxConcurrentSenders: # [number] Connection limit - Maximum number of concurrent

# -------------------------------------------------------

compression: # [string] Compression - Codec to use to compress the data before se
logFailedRequests: # [boolean] Log failed requests to disk - Use to troubleshoot
throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to throttl
tls: # [object] TLS settings (client side)
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  rejectUnauthorized: # [boolean] Validate server certs - Reject certificates tha
                trusted CA (such as the system's). Defaults to Enabled. Overrides
  servername: # [string] Server name (SNI) - Server name for the SNI (Server Name
  certificateName: # [string] Certificate - The name of the predefined certificat
  caPath: # [string] CA certificate path - Path on client in which to find CA cer
  privKeyPath: # [string] Private key path (mutual auth) - Path on client in whic
  certPath: # [string] Certificate path (mutual auth) - Path on client in which t
  passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
  minVersion: # [string] Minimum TLS version
  maxVersion: # [string] Maximum TLS version
```

```
    # ------------------------------------------------------

  connectionTimeout: # [number] Connection timeout - Amount of time (milliseconds)
  writeTimeout: # [number] Write timeout - Amount of time (milliseconds) to wait fo
  tokenTTLMinutes: # [number; minimum: 1, maximum: 60] Auth Token TTL minutes - The
  sendHeader: # [boolean] Send auth token in initial record - Upon connection, send
  onBackpressure: # [string] Backpressure behavior - How to handle events when all

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] File size limit - The maximum size to store in each que
  pqMaxSize: # [string] Queue size limit - The maximum disk space that the queue ca
  pqPath: # [string] Queue file path - The location for the persistent queue files.
  pqCompress: # [string] Compression - Codec to use to compress the persisted data
  pqOnBackpressure: # [string] Queue-full behavior - How to handle events when the
  pqMode: # [string] Mode - In Error mode, PQ writes events to the filesystem if th

  # ------------------------------------------------------

  authType: # [string] Authentication method - Select Manual to enter an auth token
  authToken: # [string] Auth token - Optional authentication token to include as pa

  # -------------- if authType is manual ---------------


  # ------------------------------------------------------

  textSecret: # [string] Auth token (text secret) - Select or create a stored text

  # -------------- if authType is secret ---------------


  # ------------------------------------------------------

  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
wavefront_output: # [object]
  type: # [string] Output Type
  authType: # [string] Authentication method - Select Manual to enter an auth token
```

```
# -------------- if authType is manual ---------------

token: # [string] Auth token - WaveFront API authentication token (see [here](htt

# -----------------------------------------------------


# -------------- if authType is secret ---------------

textSecret: # [string] Auth token (text secret) - Select or create a stored text

# -----------------------------------------------------

domain: # [string] Domain name - WaveFront domain name, e.g. "longboard"
concurrency: # [number; minimum: 1, maximum: 32] Request concurrency - Maximum nu
maxPayloadSizeKB: # [number; minimum: 1024, maximum: 10240] Body size limit (KB)
maxPayloadEvents: # [number] Events-per-request limit - Maximum number of events
compress: # [boolean] Compress - Compress the payload body before sending
rejectUnauthorized: # [boolean] Validate server certs - Reject certificates not a
    Enabled by default. When this setting is also present in TLS Settings (Client
    that value will take precedence.
timeoutSec: # [number; minimum: 1, maximum: 9007199254740991] Request timeout - A
flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
extraHttpHeaders: # [array] Extra HTTP headers - Headers to add to all events
  - name: # [string] Field Name
    value: # [string] Field Value
useRoundRobinDns: # [boolean] Round-robin DNS - Enable round-robin DNS lookup. Wh
failedRequestLoggingMode: # [string] Failed request logging mode - Data to log wh
safeHeaders: # [array of strings] Safe headers - List of headers that are safe to
responseRetrySettings: # [array] Settings for failed HTTP requests - Automaticall
  - httpStatus: # [number; minimum: 100, maximum: 599] HTTP status code - The HTT
    initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How l
    backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base fo
    maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) -
timeoutRetrySettings: # [object]
  timeoutRetry: # [boolean] Retry timed-out HTTP requests

  # -------------- if timeoutRetry is true ---------------

  initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How lon
  backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base for
  maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) - Th
```

```
    # ------------------------------------------------------

  responseHonorRetryAfterHeader: # [boolean] Honor Retry-After header - Honor any R
  onBackpressure: # [string] Backpressure behavior - How to handle events when all

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] File size limit - The maximum size to store in each que
  pqMaxSize: # [string] Queue size limit - The maximum disk space that the queue ca
  pqPath: # [string] Queue file path - The location for the persistent queue files.
  pqCompress: # [string] Compression - Codec to use to compress the persisted data
  pqOnBackpressure: # [string] Queue-full behavior - How to handle events when the
  pqMode: # [string] Mode - In Error mode, PQ writes events to the filesystem if th

  # -------------------------------------------------------

  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
signalfx_output: # [object]
  type: # [string] Output Type
  authType: # [string] Authentication method - Select Manual to enter an auth token

  # -------------- if authType is manual ---------------

  token: # [string] Auth token - SignalFx API access token (see [here](https://docs

  # -------------------------------------------------------


  # -------------- if authType is secret ---------------

  textSecret: # [string] Auth token (text secret) - Select or create a stored text

  # -------------------------------------------------------

  realm: # [string] Realm - SignalFx realm name, e.g. "us0". For a complete list of
  concurrency: # [number; minimum: 1, maximum: 32] Request concurrency - Maximum nu
  maxPayloadSizeKB: # [number; minimum: 1024, maximum: 10240] Body size limit (KB)
  maxPayloadEvents: # [number] Events-per-request limit - Maximum number of events
  compress: # [boolean] Compress - Compress the payload body before sending
```

```
rejectUnauthorized: # [boolean] Validate server certs - Reject certificates not a
    Enabled by default. When this setting is also present in TLS Settings (Client
    that value will take precedence.
timeoutSec: # [number; minimum: 1, maximum: 9007199254740991] Request timeout - A
flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
extraHttpHeaders: # [array] Extra HTTP headers - Headers to add to all events
  - name: # [string] Field Name
    value: # [string] Field Value
useRoundRobinDns: # [boolean] Round-robin DNS - Enable round-robin DNS lookup. Wh
failedRequestLoggingMode: # [string] Failed request logging mode - Data to log wh
safeHeaders: # [array of strings] Safe headers - List of headers that are safe to
responseRetrySettings: # [array] Settings for failed HTTP requests - Automaticall
  - httpStatus: # [number; minimum: 100, maximum: 599] HTTP status code - The HTT
    initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How l
    backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base fo
    maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) -
timeoutRetrySettings: # [object]
  timeoutRetry: # [boolean] Retry timed-out HTTP requests

  # -------------- if timeoutRetry is true ---------------

  initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How lon
  backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base for
  maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) - Th

  # ------------------------------------------------------

responseHonorRetryAfterHeader: # [boolean] Honor Retry-After header - Honor any R
onBackpressure: # [string] Backpressure behavior - How to handle events when all

# -------------- if onBackpressure is queue ---------------

pqMaxFileSize: # [string] File size limit - The maximum size to store in each que
pqMaxSize: # [string] Queue size limit - The maximum disk space that the queue ca
pqPath: # [string] Queue file path - The location for the persistent queue files.
pqCompress: # [string] Compression - Codec to use to compress the persisted data
pqOnBackpressure: # [string] Queue-full behavior - How to handle events when the
pqMode: # [string] Mode - In Error mode, PQ writes events to the filesystem if th

# -------------------------------------------------------

description: # [string] Description
pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
```

```
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
filesystem_output: # [object]
  type: # [string] Output Type
  destPath: # [string] Output location - Final destination for the output files
  stagePath: # [string] Staging location - Filesystem location in which to buffer f
  addIdToStagePath: # [boolean] Add output ID - Add the Output ID value to staging
  removeEmptyDirs: # [boolean] Remove empty staging directories - Remove empty stag

  # -------------- if removeEmptyDirs is true ---------------

  emptyDirCleanupSec: # [number; minimum: 10, maximum: 86400] Staging cleanup perio

  # ----------------------------------------------------------

  partitionExpr: # [string] Partitioning expression - JavaScript expression definin
  format: # [string] Data format - Format of the output data

  # -------------- if format is json ---------------

  compress: # [string] Compression - Data compression format to apply to HTTP conte
  compressionLevel: # [string] Compression level - Compression level to apply befor

  # ----------------------------------------------------------


  # -------------- if format is parquet ---------------

  automaticSchema: # [boolean] Automatic schema - Automatically calculate the schem
  parquetVersion: # [string] Parquet version - Determines which data types are supp
  parquetDataPageVersion: # [string] Data page version - Serialization format of da
  parquetRowGroupLength: # [number; minimum: 1, maximum: 67108864] Group row limit
  parquetPageSize: # [string] Page size - Target memory size for page segments, suc
  shouldLogInvalidRows: # [boolean] Log invalid rows - Log up to 3 rows that Cribl
  keyValueMetadata: # [array] Metadata (optional) - The metadata of files the Desti
    - key: # [string] Key
      value: # [string] Value
  enableStatistics: # [boolean] Write statistics - Statistics profile an entire fil
  enableWritePageIndex: # [boolean] Write page indexes - One page index contains st
  enablePageChecksum: # [boolean] Write page checksum - Parquet tools can use the c

  # ----------------------------------------------------------
```

```
  baseFileName: # [string] File name prefix expression - JavaScript expression to c
  fileNameSuffix: # [string] File name suffix expression - JavaScript expression to
  maxFileSizeMB: # [number; minimum: 5, maximum: 1024] File size limit (MB) - Maxim
  maxFileOpenTimeSec: # [number; minimum: 10, maximum: 1800] File open time limit (
  maxFileIdleTimeSec: # [number; minimum: 5, maximum: 1800] Idle time limit (sec) -
  maxOpenFiles: # [number; minimum: 10, maximum: 2000] Open file limit - Maximum nu
  headerLine: # [string] Header line - If set, this line will be written to the beg
  writeHighWaterMark: # [number; minimum: 16, maximum: 4096] Writing high watermark
  onBackpressure: # [string] Backpressure behavior - How to handle events when all
  deadletterEnabled: # [boolean] Enable dead-lettering - If a file fails to move to

  # -------------- if deadletterEnabled is true ---------------

  deadletterPath: # [string] Dead-letter location - Storage location for files that
  maxRetryNum: # [number; minimum: 1] Retry limit - The maximum number of times a f

  # ------------------------------------------------------

  onDiskFullBackpressure: # [string] Disk space protection - How to handle events w
  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
s3_output: # [object]
  type: # [string] Output Type
  bucket: # [string] S3 bucket name - Name of the destination S3 bucket. Must be a
  region: # [string] Region - Region where the S3 bucket is located
  awsSecretKey: # [string] Secret key - Secret key. This value can be a constant or
  awsAuthenticationMethod: # [string] Authentication method - AWS authentication me

  # -------------- if awsAuthenticationMethod is manual ---------------

  awsApiKey: # [string] Access key - This value can be a constant or a JavaScript e

  # ------------------------------------------------------


  # -------------- if awsAuthenticationMethod is secret ---------------

  awsSecret: # [string] Secret key pair - Select or create a stored secret that ref
```

```
# -----------------------------------------------------

endpoint: # [string] Endpoint - S3 service endpoint. If empty, defaults to the AW
signatureVersion: # [string] Signature version - Signature version to use for sig
reuseConnections: # [boolean] Reuse connections - Reuse connections between reque
rejectUnauthorized: # [boolean] Reject unauthorized certificates - Reject certifi
enableAssumeRole: # [boolean] Enable for S3 - Use Assume Role credentials to acce
assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the role
assumeRoleExternalId: # [string] External ID - External ID to use when assuming r
durationSeconds: # [number; minimum: 900, maximum: 43200] Duration (seconds) - Du
stagePath: # [string] [required] Staging location - Filesystem location in which
addIdToStagePath: # [boolean] Add output ID - Add the Output ID value to staging
destPath: # [string] Key prefix - Prefix to prepend to files before uploading. Mu
objectACL: # [string] Object ACL - Object ACL to assign to uploaded objects
storageClass: # [string] Storage class - Storage class to select for uploaded obj
serverSideEncryption: # [string] Server-side encryption for uploaded objects

# -------------- if serverSideEncryption is aws:kms ---------------

kmsKeyId: # [string] KMS key ID - ID or ARN of the KMS customer-managed key to us

# -----------------------------------------------------

removeEmptyDirs: # [boolean] Remove empty staging directories - Remove empty stag

# -------------- if removeEmptyDirs is true ---------------

emptyDirCleanupSec: # [number; minimum: 10, maximum: 86400] Staging cleanup perio

# -----------------------------------------------------

partitionExpr: # [string] Partitioning expression - JavaScript expression definin
format: # [string] Data format - Format of the output data

# -------------- if format is json ---------------

compress: # [string] Compression - Data compression format to apply to HTTP conte
compressionLevel: # [string] Compression level - Compression level to apply befor

# -----------------------------------------------------


# -------------- if format is parquet ---------------
```

```
automaticSchema: # [boolean] Automatic schema - Automatically calculate the schem
parquetVersion: # [string] Parquet version - Determines which data types are supp
parquetDataPageVersion: # [string] Data page version - Serialization format of da
parquetRowGroupLength: # [number; minimum: 1, maximum: 67108864] Group row limit
parquetPageSize: # [string] Page size - Target memory size for page segments, suc
shouldLogInvalidRows: # [boolean] Log invalid rows - Log up to 3 rows that Cribl
keyValueMetadata: # [array] Metadata (optional) - The metadata of files the Desti
  - key: # [string] Key
    value: # [string] Value
enableStatistics: # [boolean] Write statistics - Statistics profile an entire fil
enableWritePageIndex: # [boolean] Write page indexes - One page index contains st
enablePageChecksum: # [boolean] Write page checksum - Parquet tools can use the c

# ------------------------------------------------------

baseFileName: # [string] File name prefix expression - JavaScript expression to c
fileNameSuffix: # [string] File name suffix expression - JavaScript expression to
maxFileSizeMB: # [number; minimum: 5, maximum: 1024] File size limit (MB) - Maxim
maxOpenFiles: # [number; minimum: 10, maximum: 2000] Open file limit - Maximum nu
headerLine: # [string] Header line - If set, this line will be written to the beg
writeHighWaterMark: # [number; minimum: 16, maximum: 4096] Writing high watermark
onBackpressure: # [string] Backpressure behavior - How to handle events when all
deadletterEnabled: # [boolean] Enable dead-lettering - If a file fails to move to

# -------------- if deadletterEnabled is true ---------------

deadletterPath: # [string] Dead-letter location - Storage location for files that
maxRetryNum: # [number; minimum: 1] Retry limit - The maximum number of times a f

# ------------------------------------------------------

onDiskFullBackpressure: # [string] Disk space protection - How to handle events w
maxFileOpenTimeSec: # [number; minimum: 10, maximum: 86400] File open time limit
maxFileIdleTimeSec: # [number; minimum: 5, maximum: 86400] Idle time limit (sec)
maxConcurrentFileParts: # [number; minimum: 1, maximum: 10] Concurrent file parts
verifyPermissions: # [boolean] Verify if bucket exists - Disable if you can acces
maxClosingFilesToBackpressure: # [number; minimum: 10, maximum: 4200] Staging fil
description: # [string] Description
pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
systemFields: # [array of strings] System fields - Fields to automatically add to
environment: # [string] Environment - Optionally, enable this config only on a sp
streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
```

```
azure_blob_output: # [object]
  type: # [string] Output Type
  containerName: # [string] Container name - The Azure Blob Storage container name.
  createContainer: # [boolean] Create container - Create the configured container i
  destPath: # [string] Blob prefix - Root directory prepended to path before uploac
  stagePath: # [string] [required] Staging location - Filesystem location in which
  addIdToStagePath: # [boolean] Add output ID - Add the Output ID value to staging
  maxConcurrentFileParts: # [number; minimum: 1, maximum: 10] Concurrent file parts
  removeEmptyDirs: # [boolean] Remove empty staging directories - Remove empty stag

  # -------------- if removeEmptyDirs is true ---------------

  emptyDirCleanupSec: # [number; minimum: 10, maximum: 86400] Staging cleanup perio

  # ----------------------------------------------------------

  partitionExpr: # [string] Partitioning expression - JavaScript expression definir
  format: # [string] Data format - Format of the output data

  # -------------- if format is json ---------------

  compress: # [string] Compression - Data compression format to apply to HTTP conte
  compressionLevel: # [string] Compression level - Compression level to apply befor

  # ----------------------------------------------------------


  # -------------- if format is parquet ---------------

  automaticSchema: # [boolean] Automatic schema - Automatically calculate the schem
  parquetVersion: # [string] Parquet version - Determines which data types are supp
  parquetDataPageVersion: # [string] Data page version - Serialization format of da
  parquetRowGroupLength: # [number; minimum: 1, maximum: 67108864] Group row limit
  parquetPageSize: # [string] Page size - Target memory size for page segments, suc
  shouldLogInvalidRows: # [boolean] Log invalid rows - Log up to 3 rows that Cribl
  keyValueMetadata: # [array] Metadata (optional) - The metadata of files the Desti
    - key: # [string] Key
      value: # [string] Value
  enableStatistics: # [boolean] Write statistics - Statistics profile an entire fil
  enableWritePageIndex: # [boolean] Write page indexes - One page index contains st
  enablePageChecksum: # [boolean] Write page checksum - Parquet tools can use the c

  # ----------------------------------------------------------
```

```
baseFileName: # [string] File name prefix expression - JavaScript expression to c
fileNameSuffix: # [string] File name suffix expression - JavaScript expression to
maxFileSizeMB: # [number; minimum: 5, maximum: 1024] File size limit (MB) - Maxim
maxFileOpenTimeSec: # [number; minimum: 10, maximum: 1800] File open time limit (
maxFileIdleTimeSec: # [number; minimum: 5, maximum: 1800] Idle time limit (sec) -
maxOpenFiles: # [number; minimum: 10, maximum: 2000] Open file limit - Maximum nu
headerLine: # [string] Header line - If set, this line will be written to the beg
writeHighWaterMark: # [number; minimum: 16, maximum: 4096] Writing high watermark
onBackpressure: # [string] Backpressure behavior - How to handle events when all
deadletterEnabled: # [boolean] Enable dead-lettering - If a file fails to move to

# -------------- if deadletterEnabled is true ---------------

deadletterPath: # [string] Dead-letter location - Storage location for files that
maxRetryNum: # [number; minimum: 1] Retry limit - The maximum number of times a f

# ------------------------------------------------------------

onDiskFullBackpressure: # [string] Disk space protection - How to handle events w
authType: # [string] Authentication method

# -------------- if authType is manual ---------------

connectionString: # [string] Connection string - Enter your Azure Storage account

# ------------------------------------------------------------


# -------------- if authType is secret ---------------

textSecret: # [string] Connection string (text secret) - Select or create a store

# ------------------------------------------------------------


# -------------- if authType is clientSecret ---------------

storageAccountName: # [string] Storage account name - The name of your Azure stor
tenantId: # [string] Tenant ID - The service principal's tenant ID
clientId: # [string] Client ID - The service principal's client ID
azureCloud: # [string] Azure Cloud - The Azure cloud to use. Defaults to Azure Pu
endpointSuffix: # [string] Endpoint suffix - Endpoint suffix for the service URL.
```

```
  clientTextSecret: # [string] Client secret (text secret) - Select or create a stc

  # ------------------------------------------------------


  # -------------- if authType is clientCert ---------------

  storageAccountName: # [string] Storage account name - The name of your Azure stor
  tenantId: # [string] Tenant ID - The service principal's tenant ID
  clientId: # [string] Client ID - The service principal's client ID
  azureCloud: # [string] Azure Cloud - The Azure cloud to use. Defaults to Azure Pu
  endpointSuffix: # [string] Endpoint suffix - Endpoint suffix for the service URL.
  certificate: # [object]
    certificateName: # [string] Certificate - The certificate you registered as cre

  # ------------------------------------------------------

  storageClass: # [string] Blob access tier
  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
azure_data_explorer_output: # [object]
  type: # [string] Output Type
  clusterUrl: # [string] Cluster base URI - The base URI for your cluster. Typicall
  database: # [string] [required] Database name - Name of the database containing t
  table: # [string] [required] Table name - Name of the table to ingest data into
  validateDatabaseSettings: # [boolean] Validate database settings - When saving or
  ingestMode: # [string] Ingestion mode

  # -------------- if ingestMode is batching ---------------

  ingestUrl: # [string] Ingestion service URI - The ingestion service URI for your
  onBackpressure: # [string] Backpressure behavior - How to handle events when all
  isMappingObj: # [boolean] Add mapping object - Send a JSON mapping object instead
  format: # [string] Data format - Format of the output data
  stagePath: # [string] Staging location - Filesystem location in which to buffer f
  fileNameSuffix: # [string] File name suffix expression - JavaScript expression to
  maxFileSizeMB: # [number; minimum: 5, maximum: 1024] File size limit (MB) - Maxim
  maxFileOpenTimeSec: # [number; minimum: 10, maximum: 1800] File open time limit (
  maxFileIdleTimeSec: # [number; minimum: 5, maximum: 1800] Idle time limit (sec) -
  maxOpenFiles: # [number; minimum: 10, maximum: 2000] Open file limit - Maximum nu
```

```
maxConcurrentFileParts: # [number; minimum: 1, maximum: 10] Concurrent file parts
onDiskFullBackpressure: # [string] Disk space protection - How to handle events w
addIdToStagePath: # [boolean] Add output ID - Add the Output ID value to staging
removeEmptyDirs: # [boolean] Remove empty staging directories - Remove empty stag
deadletterEnabled: # [boolean] Enable dead-lettering - If a file fails to move to
timeoutSec: # [number; minimum: 1, maximum: 9007199254740991] Request timeout - A
flushImmediately: # [boolean] Flush immediately - Bypass the data management serv
retainBlobOnSuccess: # [boolean] Retain blob on success - Prevent blob deletion a
extentTags: # [array] Extent tags - Strings or tags associated with the extent (i
  - prefix: # [string] Prefix (optional)
    value: # [string] Value
ingestIfNotExists: # [array] Enforce uniqueness via tag values - Prevents duplica
  - value: # [string] Value
reportLevel: # [string] Report level - Level of ingestion status reporting. Defau
reportMethod: # [string] Report method - Target of the ingestion status reporting
additionalProperties: # [array] Additional fields - Optionally, enter additional
  - key: # [string] Key
    value: # [string] Value

# ------------------------------------------------------


# -------------- if ingestMode is streaming ---------------

responseRetrySettings: # [array] Settings for failed HTTP requests - Automaticall
  - httpStatus: # [number; minimum: 100, maximum: 599] HTTP status code - The HTT
    initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How l
    backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base fo
    maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) -
timeoutRetrySettings: # [object]
  timeoutRetry: # [boolean] Retry timed-out HTTP requests

  # -------------- if timeoutRetry is true ---------------

  initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How lon
  backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base for
  maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) - Th

  # ------------------------------------------------------

responseHonorRetryAfterHeader: # [boolean] Honor Retry-After header - Honor any R
onBackpressure: # [string] Backpressure behavior - How to handle events when all
format: # [string] Data format - Format of the output data
```

```
compress: # [string] [required] Compression - Data compression format to apply to
mappingRef: # [string] Data mapping - Enter the name of a data mapping associated
timeoutSec: # [number; minimum: 1, maximum: 9007199254740991] Request timeout - A
concurrency: # [number; minimum: 1, maximum: 32] Request concurrency - Maximum nu
maxPayloadSizeKB: # [number; minimum: 1024, maximum: 4096] Body size limit (KB) -
maxPayloadEvents: # [number] Events-per-request limit - Maximum number of events
flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
rejectUnauthorized: # [boolean] Validate server certs - Reject certificates not a
    Enabled by default. When this setting is also present in TLS Settings (Client
    that value will take precedence.
useRoundRobinDns: # [boolean] Round-robin DNS - Enable round-robin DNS lookup. Wh
keepAlive: # [boolean] Keep alive - Disable to close the connection immediately a

# ------------------------------------------------------

oauthEndpoint: # [string] [required] Microsoft Entra ID authentication endpoint -
tenantId: # [string] [required] Tenant ID - Directory ID (tenant identifier) in A
clientId: # [string] [required] Client ID - client_id to pass in the OAuth reques
scope: # [string] [required] Scope - Scope to pass in the OAuth request parameter
oauthType: # [string] [required] Authentication method - The type of OAuth 2.0 cl

# -------------- if oauthType is clientSecret ---------------

clientSecret: # [string] Client secret - The client secret that you generated for

# ------------------------------------------------------


# -------------- if oauthType is clientTextSecret ---------------

textSecret: # [string] Client secret (text secret) - Select or create a stored te

# ------------------------------------------------------


# -------------- if oauthType is certificate ---------------

certificate: # [object]
  certificateName: # [string] Certificate - The certificate you registered as cre

# ------------------------------------------------------

advancedSettingsNote: # [null]
```

```
  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
azure_logs_output: # [object]
  type: # [string] Output Type
  logType: # [string] Log Type - The Log Type of events sent to this LogAnalytics w
  resourceId: # [string] Resource ID - Optional Resource ID of the Azure resource t
  concurrency: # [number; minimum: 1, maximum: 32] Request concurrency - Maximum nu
  maxPayloadSizeKB: # [number; minimum: 1, maximum: 10240] Body size limit (KB) - M
  maxPayloadEvents: # [number] Events-per-request limit - Maximum number of events
  compress: # [boolean]
  rejectUnauthorized: # [boolean] Validate server certs - Reject certificates not a
      Enabled by default. When this setting is also present in TLS Settings (Client
      that value will take precedence.
  timeoutSec: # [number; minimum: 1, maximum: 9007199254740991] Request timeout - A
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
  extraHttpHeaders: # [array] Extra HTTP headers - Headers to add to all events
    - name: # [string] Field Name
      value: # [string] Field Value
  useRoundRobinDns: # [boolean] Round-robin DNS - Enable round-robin DNS lookup. Wh
  failedRequestLoggingMode: # [string] Failed request logging mode - Data to log wh
  safeHeaders: # [array of strings] Safe headers - List of headers that are safe to
  apiUrl: # [string] DNS name of API endpoint - The DNS name of the Log API endpoin
  responseRetrySettings: # [array] Settings for failed HTTP requests - Automatical
    - httpStatus: # [number; minimum: 100, maximum: 599] HTTP status code - The HTT
      initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How l
      backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base fo
      maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) -
  timeoutRetrySettings: # [object]
    timeoutRetry: # [boolean] Retry timed-out HTTP requests

    # -------------- if timeoutRetry is true ---------------

    initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How lon
    backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base for
    maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) - Th

    # -----------------------------------------------------

  responseHonorRetryAfterHeader: # [boolean] Honor Retry-After header - Honor any R
  onBackpressure: # [string] Backpressure behavior - How to handle events when all
```

```
  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] File size limit - The maximum size to store in each que
  pqMaxSize: # [string] Queue size limit - The maximum disk space that the queue ca
  pqPath: # [string] Queue file path - The location for the persistent queue files.
  pqCompress: # [string] Compression - Codec to use to compress the persisted data
  pqOnBackpressure: # [string] Queue-full behavior - How to handle events when the
  pqMode: # [string] Mode - In Error mode, PQ writes events to the filesystem if th

  # ---------------------------------------------------------

  authType: # [string] Authentication method - Enter workspace ID and workspace key
  workspaceId: # [string] Workspace ID - Azure Log Analytics Workspace ID. See Azur
  workspaceKey: # [string] Workspace key - Azure Log Analytics Workspace Primary or

  # -------------- if authType is manual ---------------


  # ---------------------------------------------------------

  keypairSecret: # [string] Secret key pair - Select or create a stored secret that

  # -------------- if authType is secret ---------------


  # ---------------------------------------------------------

  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
kinesis_output: # [object]
  type: # [string] Output Type
  streamName: # [string] Stream Name - Kinesis stream name to send events to.
  awsAuthenticationMethod: # [string] Authentication method - AWS authentication me

  # -------------- if awsAuthenticationMethod is manual ---------------

  awsApiKey: # [string] Access key

  # ---------------------------------------------------------
```

```
# -------------- if awsAuthenticationMethod is secret ---------------

awsSecret: # [string] Secret key pair - Select or create a stored secret that ref

# ------------------------------------------------------

awsSecretKey: # [string] Secret key
region: # [string] [required] Region - Region where the Kinesis stream is located
endpoint: # [string] Endpoint - Kinesis stream service endpoint. If empty, defaul
signatureVersion: # [string] Signature version - Signature version to use for sig
reuseConnections: # [boolean] Reuse connections - Reuse connections between reque
rejectUnauthorized: # [boolean] Reject unauthorized certificates - Reject certifi
enableAssumeRole: # [boolean] Enable for Kinesis stream - Use Assume Role credent
assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the role
assumeRoleExternalId: # [string] External ID - External ID to use when assuming r
durationSeconds: # [number; minimum: 900, maximum: 43200] Duration (seconds) - Du
concurrency: # [number; minimum: 1, maximum: 32] Put request concurrency - Maximu
maxRecordSizeKB: # [number; minimum: 1, maximum: 10240] Record size limit (KB, ur
flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
compression: # [string] Compression - Compression type to use for records

# -------------- if compression is none ---------------

asNdjson: # [boolean] Send batched - Batch events into a single record as NDJSON

# ------------------------------------------------------

useListShards: # [boolean] ListShards API - Provides higher stream rate limits, i
onBackpressure: # [string] Backpressure behavior - How to handle events when all

# -------------- if onBackpressure is queue ---------------

pqMaxFileSize: # [string] File size limit - The maximum size to store in each que
pqMaxSize: # [string] Queue size limit - The maximum disk space that the queue ca
pqPath: # [string] Queue file path - The location for the persistent queue files.
pqCompress: # [string] Compression - Codec to use to compress the persisted data
pqOnBackpressure: # [string] Queue-full behavior - How to handle events when the
pqMode: # [string] Mode - In Error mode, PQ writes events to the filesystem if th

# ------------------------------------------------------
```

```
  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
honeycomb_output: # [object]
  type: # [string] Output Type
  dataset: # [string] Dataset name - Name of the dataset to send events to - e.g.,
  concurrency: # [number; minimum: 1, maximum: 32] Request concurrency - Maximum nu
  maxPayloadSizeKB: # [number; minimum: 1024, maximum: 10240] Body size limit (KB)
  maxPayloadEvents: # [number] Events-per-request limit - Maximum number of events
  compress: # [boolean] Compress - Compress the payload body before sending
  rejectUnauthorized: # [boolean] Validate server certs - Reject certificates not a
      Enabled by default. When this setting is also present in TLS Settings (Client
      that value will take precedence.
  timeoutSec: # [number; minimum: 1, maximum: 9007199254740991] Request timeout - A
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
  extraHttpHeaders: # [array] Extra HTTP headers - Headers to add to all events
    - name: # [string] Field Name
      value: # [string] Field Value
  useRoundRobinDns: # [boolean] Round-robin DNS - Enable round-robin DNS lookup. Wh
  failedRequestLoggingMode: # [string] Failed request logging mode - Data to log wh
  safeHeaders: # [array of strings] Safe headers - List of headers that are safe to
  responseRetrySettings: # [array] Settings for failed HTTP requests - Automatical
    - httpStatus: # [number; minimum: 100, maximum: 599] HTTP status code - The HTT
      initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How l
      backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base fo
      maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) -
  timeoutRetrySettings: # [object]
    timeoutRetry: # [boolean] Retry timed-out HTTP requests

    # -------------- if timeoutRetry is true ---------------

    initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How lon
    backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base for
    maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) - Th

    # -------------------------------------------------------

  responseHonorRetryAfterHeader: # [boolean] Honor Retry-After header - Honor any R
  onBackpressure: # [string] Backpressure behavior - How to handle events when all

  # -------------- if onBackpressure is queue ---------------
```

```
  pqMaxFileSize: # [string] File size limit - The maximum size to store in each que
  pqMaxSize: # [string] Queue size limit - The maximum disk space that the queue ca
  pqPath: # [string] Queue file path - The location for the persistent queue files.
  pqCompress: # [string] Compression - Codec to use to compress the persisted data
  pqOnBackpressure: # [string] Queue-full behavior - How to handle events when the
  pqMode: # [string] Mode - In Error mode, PQ writes events to the filesystem if th

  # ------------------------------------------------------

  authType: # [string] Authentication method - Enter API key directly, or select a
  team: # [string] API key - Team API key where the dataset belongs

  # -------------- if authType is manual ---------------


  # ------------------------------------------------------

  textSecret: # [string] API key (text secret) - Select or create a stored text sec

  # -------------- if authType is secret ---------------


  # ------------------------------------------------------

  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
azure_eventhub_output: # [object]
  type: # [string] Output Type
  brokers: # [array of strings] Brokers - List of Event Hubs Kafka brokers to conne
  topic: # [string] [required] Event Hub name - The name of the Event Hub (Kafka To
  ack: # [integer] Acknowledgments - Control the number of required acknowledgments
  format: # [string] Record data format - Format to use to serialize events before
  maxRecordSizeKB: # [number; minimum: 1] Record size limit (KB, uncompressed) - Ma
  flushEventCount: # [number; minimum: 1, maximum: 10000] Events-per-batch limit -
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
  connectionTimeout: # [number; minimum: 1000, maximum: 3600000] Connection timeout
  requestTimeout: # [number; minimum: 1000, maximum: 3600000] Request timeout (ms)
  maxRetries: # [number; maximum: 100] Retry limit - If messages are failing, you c
  maxBackOff: # [number; minimum: 30000, maximum: 180000] Backoff limit (ms) - The
```

```
  initialBackoff: # [number; minimum: 300, maximum: 600000] Initial retry interval
  backoffRate: # [number; minimum: 2, maximum: 20] Backoff multiplier - Set the bac
  authenticationTimeout: # [number; minimum: 1000, maximum: 3600000] Authentication
  reauthenticationThreshold: # [number; minimum: 1000, maximum: 1800000] Reauthenti
  sasl: # [object] Authentication - Authentication parameters to use when connectin
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    mechanism: # [string] SASL mechanism

    # ---------------------------------------------------

  tls: # [object] TLS settings (client side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    rejectUnauthorized: # [boolean] Validate server certs - Reject certificates tha

    # ---------------------------------------------------

  onBackpressure: # [string] Backpressure behavior - How to handle events when all

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] File size limit - The maximum size to store in each que
  pqMaxSize: # [string] Queue size limit - The maximum disk space that the queue ca
  pqPath: # [string] Queue file path - The location for the persistent queue files.
  pqCompress: # [string] Compression - Codec to use to compress the persisted data
  pqOnBackpressure: # [string] Queue-full behavior - How to handle events when the
  pqMode: # [string] Mode - In Error mode, PQ writes events to the filesystem if th

  # ---------------------------------------------------

  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
google_chronicle_output: # [object]
  type: # [string] Output Type
  apiVersion: # [string] API version
```

```
# -------------- if apiVersion is v1 ---------------

authenticationMethod: # [string] Authentication method

# --------------------------------------------------------

# -------------- if apiVersion is v2 ---------------

authenticationMethod: # [string] Authentication method

# --------------------------------------------------------

authenticationMethod: # [string] Authentication method

# -------------- if authenticationMethod is manual ---------------

apiKey: # [string] API key - Organization's API key in Google SecOps

# --------------------------------------------------------

# -------------- if authenticationMethod is secret ---------------

apiKeySecret: # [string] API key (text secret) - Select or create a stored text s

# --------------------------------------------------------

# -------------- if authenticationMethod is serviceAccount ---------------

serviceAccountCredentials: # [string] Service account credentials - Contents of s

# --------------------------------------------------------

# -------------- if authenticationMethod is serviceAccountSecret ---------------

serviceAccountCredentialsSecret: # [string] Service account credentials (text sec

# --------------------------------------------------------
```

```
responseRetrySettings: # [array] Settings for failed HTTP requests - Automaticall
  - httpStatus: # [number; minimum: 100, maximum: 599] HTTP status code - The HT
    initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How l
    backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base fo
    maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) -
timeoutRetrySettings: # [object]
  timeoutRetry: # [boolean] Retry timed-out HTTP requests

  # -------------- if timeoutRetry is true ---------------

  initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How lon
  backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base for
  maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) - Th

  # ------------------------------------------------------

responseHonorRetryAfterHeader: # [boolean] Honor Retry-After header - Honor any R
logFormatType: # [string] Send events as

# -------------- if logFormatType is unstructured ---------------

extraLogTypes: # [array] Custom log types - Custom log types. If the value "Custo
  - logType: # [string] Log Type
    description: # [string] Description
logType: # [string] Default log type - Default log type value to send to SecOps.
logTextField: # [string] Log text field - Name of the event field that contains t
customerId: # [string] Customer ID - A unique identifier (UUID) for your Google S
namespace: # [string] Namespace - User-configured environment namespace to identi
customLabels: # [array] Custom labels - Custom labels to be added to every batch
  - key: # [string] Key
    value: # [string] Value

# ------------------------------------------------------

region: # [string] Region - Regional endpoint to send events to
concurrency: # [number; minimum: 1, maximum: 32] Request concurrency - Maximum nu
maxPayloadSizeKB: # [number; minimum: 1, maximum: 1024] Body size limit (KB) - Ma
maxPayloadEvents: # [number] Events-per-request limit - Maximum number of events
compress: # [boolean] Compress - Compress the payload body before sending
rejectUnauthorized: # [boolean] Validate server certs - Reject certificates not a
    Enabled by default. When this setting is also present in TLS Settings (Client
    that value will take precedence.
timeoutSec: # [number; minimum: 1, maximum: 9007199254740991] Request timeout - A
```

```
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
  extraHttpHeaders: # [array] Extra HTTP headers - Headers to add to all events
    - name: # [string] Field Name
      value: # [string] Field Value
  failedRequestLoggingMode: # [string] Failed request logging mode - Data to log wh
  safeHeaders: # [array of strings] Safe headers - List of headers that are safe to
  useRoundRobinDns: # [boolean] Round-robin DNS - Enable round-robin DNS lookup. Wh
  onBackpressure: # [string] Backpressure behavior - How to handle events when all

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] File size limit - The maximum size to store in each que
  pqMaxSize: # [string] Queue size limit - The maximum disk space that the queue ca
  pqPath: # [string] Queue file path - The location for the persistent queue files.
  pqCompress: # [string] Compression - Codec to use to compress the persisted data
  pqOnBackpressure: # [string] Queue-full behavior - How to handle events when the
  pqMode: # [string] Mode - In Error mode, PQ writes events to the filesystem if th

  # ---------------------------------------------------------

  totalMemoryLimitKB: # [number] Buffer memory limit (KB) - Maximum total size of t
  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
google_cloud_storage_output: # [object]
  type: # [string] Output Type
  bucket: # [string] Bucket name - Name of the destination bucket. This value can b
  region: # [string] [required] Region - Region where the bucket is located
  endpoint: # [string] [required] Endpoint - Google Cloud Storage service endpoint
  signatureVersion: # [string] Signature version - Signature version to use for sig
  awsAuthenticationMethod: # [string] Authentication method

  # -------------- if awsAuthenticationMethod is manual ---------------

  awsApiKey: # [string] Access key - HMAC access key. This value can be a constant
  awsSecretKey: # [string] Secret - HMAC secret. This value can be a constant or a

  # ---------------------------------------------------------


  # -------------- if awsAuthenticationMethod is secret ---------------
```

```
awsSecret: # [string] Secret key pair - Select or create a stored secret that ref

# ------------------------------------------------------

stagePath: # [string] [required] Staging location - Filesystem location in which
destPath: # [string] Key prefix - Prefix to prepend to files before uploading. Mu
verifyPermissions: # [boolean] Verify if bucket exists - Disable if you can acces
objectACL: # [string] Object ACL - Object ACL to assign to uploaded objects
storageClass: # [string] Storage class - Storage class to select for uploaded obj
reuseConnections: # [boolean] Reuse connections - Reuse connections between reque
rejectUnauthorized: # [boolean] Reject unauthorized certificates - Reject certifi
addIdToStagePath: # [boolean] Add output ID - Add the Output ID value to staging
removeEmptyDirs: # [boolean] Remove empty staging directories - Remove empty stag

# -------------- if removeEmptyDirs is true ---------------

emptyDirCleanupSec: # [number; minimum: 10, maximum: 86400] Staging cleanup perio

# ------------------------------------------------------

partitionExpr: # [string] Partitioning expression - JavaScript expression definin
format: # [string] Data format - Format of the output data

# -------------- if format is json ---------------

compress: # [string] Compression - Data compression format to apply to HTTP conte
compressionLevel: # [string] Compression level - Compression level to apply befor

# ------------------------------------------------------


# -------------- if format is parquet ---------------

automaticSchema: # [boolean] Automatic schema - Automatically calculate the schem
parquetVersion: # [string] Parquet version - Determines which data types are supp
parquetDataPageVersion: # [string] Data page version - Serialization format of da
parquetRowGroupLength: # [number; minimum: 1, maximum: 67108864] Group row limit
parquetPageSize: # [string] Page size - Target memory size for page segments, suc
shouldLogInvalidRows: # [boolean] Log invalid rows - Log up to 3 rows that Cribl
keyValueMetadata: # [array] Metadata (optional) - The metadata of files the Desti
  - key: # [string] Key
    value: # [string] Value
```

```
  enableStatistics: # [boolean] Write statistics - Statistics profile an entire fil
  enableWritePageIndex: # [boolean] Write page indexes - One page index contains st
  enablePageChecksum: # [boolean] Write page checksum - Parquet tools can use the c

  # ------------------------------------------------------

  baseFileName: # [string] File name prefix expression - JavaScript expression to c
  fileNameSuffix: # [string] File name suffix expression - JavaScript expression to
  maxFileSizeMB: # [number; minimum: 5, maximum: 1024] File size limit (MB) - Maxim
  maxFileOpenTimeSec: # [number; minimum: 10, maximum: 1800] File open time limit (
  maxFileIdleTimeSec: # [number; minimum: 5, maximum: 1800] Idle time limit (sec) -
  maxOpenFiles: # [number; minimum: 10, maximum: 2000] Open file limit - Maximum nu
  headerLine: # [string] Header line - If set, this line will be written to the beg
  writeHighWaterMark: # [number; minimum: 16, maximum: 4096] Writing high watermark
  onBackpressure: # [string] Backpressure behavior - How to handle events when all
  deadletterEnabled: # [boolean] Enable dead-lettering - If a file fails to move to

  # -------------- if deadletterEnabled is true ---------------

  deadletterPath: # [string] Dead-letter location - Storage location for files that
  maxRetryNum: # [number; minimum: 1] Retry limit - The maximum number of times a f

  # ------------------------------------------------------

  onDiskFullBackpressure: # [string] Disk space protection - How to handle events w
  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
google_cloud_logging_output: # [object]
  type: # [string] Output Type
  logLocationType: # [string] Log location type

  # -------------- if logLocationType is project ---------------

  logLocationExpression: # [string] [required] Project ID expression - JavaScript e

  # ------------------------------------------------------


  # -------------- if logLocationType is organization ---------------
```

```
logLocationExpression: # [string] [required] Organization ID expression - JavaScr

# ---------------------------------------------------------


# -------------- if logLocationType is billingAccount ---------------

logLocationExpression: # [string] [required] Billing account ID expression - Java

# ---------------------------------------------------------


# -------------- if logLocationType is folder ---------------

logLocationExpression: # [string] [required] Folder ID expression - JavaScript ex

# ---------------------------------------------------------

logNameExpression: # [string] [required] Log name expression - JavaScript express
payloadFormat: # [string] Payload format - Format to use when sending payload. De

# -------------- if payloadFormat is text ---------------

payloadExpression: # [string] Payload text expression - JavaScript expression to

# ---------------------------------------------------------


# -------------- if payloadFormat is json ---------------

payloadExpression: # [string] Payload object expression - JavaScript expression t

# ---------------------------------------------------------

logLabels: # [array] Log labels - Labels to apply to the log entry
  - label: # [string] Label - Label name
    valueExpression: # [string] Value - JavaScript expression to compute the labe
resourceTypeExpression: # [string] Resource type expression - JavaScript expressi
resourceTypeLabels: # [array] Resource labels - Labels to apply to the managed re
  - label: # [string] Label - Label name
    valueExpression: # [string] Value - JavaScript expression to compute the labe
severityExpression: # [string] Severity expression - JavaScript expression to com
insertIdExpression: # [string] Insert ID expression - JavaScript expression to co
```

```
googleAuthMethod: # [string] Google authentication method - Choose Auto to use Go

# -------------- if googleAuthMethod is manual ---------------

serviceAccountCredentials: # [string] Service account credentials - Contents of s

# ------------------------------------------------------------


# -------------- if googleAuthMethod is secret ---------------

secret: # [string] Service account credentials (text secret) - Select or create a

# ------------------------------------------------------------

maxPayloadSizeKB: # [number; minimum: 1024, maximum: 10240] Body size limit (KB)
maxPayloadEvents: # [number] Events-per-request limit - Max number of events to i
flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
concurrency: # [number; minimum: 1, maximum: 32] Request concurrency - Maximum nu
connectionTimeout: # [number] Connection timeout - Amount of time (milliseconds)
timeoutSec: # [number; minimum: 1, maximum: 9007199254740991] Request timeout - A
throttleRateReqPerSec: # [integer; maximum: 2000] Throttle request rate - Maximum
requestMethodExpression: # [string] Request method expression - A JavaScript expr
requestUrlExpression: # [string] Request URL expression - A JavaScript expressior
requestSizeExpression: # [string] Request size expression - A JavaScript expressi
statusExpression: # [string] Request status expression - A JavaScript expression
responseSizeExpression: # [string] Response size expression - A JavaScript expres
userAgentExpression: # [string] Request user agent expression - A JavaScript expr
remoteIpExpression: # [string] Remote IP expression - A JavaScript expression tha
serverIpExpression: # [string] Server IP expression - A JavaScript expression tha
refererExpression: # [string] Referer expression - A JavaScript expression that e
latencyExpression: # [string] Latency expression - A JavaScript expression that e
cacheLookupExpression: # [string] Cache lookup expression - A JavaScript expressi
cacheHitExpression: # [string] Cache hit expression - A JavaScript expression tha
cacheValidatedExpression: # [string] Cache validated with origin server expressio
cacheFillBytesExpression: # [string] Cache fill bytes expression - A JavaScript e
protocolExpression: # [string] Protocol expression - A JavaScript expression that
idExpression: # [string] ID expression - A JavaScript expression that evaluates t
producerExpression: # [string] Producer expression - A JavaScript expression that
firstExpression: # [string] First expression - A JavaScript expression that evalu
lastExpression: # [string] Last expression - A JavaScript expression that evaluat
fileExpression: # [string] File expression - A JavaScript expression that evaluat
lineExpression: # [string] Line expression - A JavaScript expression that evaluat
```

```
  functionExpression: # [string] Function expression - A JavaScript expression that
  uidExpression: # [string] UID expression - A JavaScript expression that evaluates
  indexExpression: # [string] Index expression - A JavaScript expression that evalu
  totalSplitsExpression: # [string] Total splits expression - A JavaScript expressi
  traceExpression: # [string] Trace expression - A JavaScript expression that evalu
  spanIdExpression: # [string] Span ID expression - A JavaScript expression that ev
  traceSampledExpression: # [string] Trace sampled expression - A JavaScript expres
  onBackpressure: # [string] Backpressure behavior - How to handle events when all

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] File size limit - The maximum size to store in each que
  pqMaxSize: # [string] Queue size limit - The maximum disk space that the queue ca
  pqPath: # [string] Queue file path - The location for the persistent queue files.
  pqCompress: # [string] Compression - Codec to use to compress the persisted data
  pqOnBackpressure: # [string] Queue-full behavior - How to handle events when the
  pqMode: # [string] Mode - In Error mode, PQ writes events to the filesystem if th

  # ---------------------------------------------------------

  totalMemoryLimitKB: # [number] Buffer memory limit (KB) - Maximum total size of t
  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
google_pubsub_output: # [object]
  type: # [string] Output Type
  topicName: # [string] Topic ID - ID of the topic to send events to.
  createTopic: # [boolean] Create topic - If enabled, create topic if it does not e
  orderedDelivery: # [boolean] Ordered delivery - If enabled, send events in the or
  region: # [string] Region - Region to publish messages to. Select 'default' to al
  googleAuthMethod: # [string] Google authentication method - Choose Auto to use Go

  # -------------- if googleAuthMethod is manual ---------------

  serviceAccountCredentials: # [string] Service account credentials - Contents of s

  # ---------------------------------------------------------


  # -------------- if googleAuthMethod is secret ---------------
```

```
  secret: # [string] Service account credentials (text secret) - Select or create a

  # ------------------------------------------------------

  batchSize: # [number; minimum: 1, maximum: 10000] Batch size - The maximum number
  batchTimeout: # [number; minimum: 1, maximum: 100000] Batch timeout (ms) - The ma
  maxQueueSize: # [number; minimum: 1] Queue size limit - Maximum number of queued
  maxRecordSizeKB: # [number; minimum: 1, maximum: 256] Batch size limit (KB) - Max
  flushPeriod: # [number] Flush period (sec) - Maximum time to wait before sending
  maxInProgress: # [number; minimum: 1, maximum: 100] Concurrent request limit - Th
  onBackpressure: # [string] Backpressure behavior - How to handle events when all

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] File size limit - The maximum size to store in each que
  pqMaxSize: # [string] Queue size limit - The maximum disk space that the queue ca
  pqPath: # [string] Queue file path - The location for the persistent queue files.
  pqCompress: # [string] Compression - Codec to use to compress the persisted data
  pqOnBackpressure: # [string] Queue-full behavior - How to handle events when the
  pqMode: # [string] Mode - In Error mode, PQ writes events to the filesystem if th

  # ------------------------------------------------------

  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
exabeam_output: # [object]
  bucket: # [string] Bucket name - Name of the destination bucket. A constant or a
  region: # [string] [required] Region - Region where the bucket is located
  stagePath: # [string] [required] Staging location - Filesystem location in which
  endpoint: # [string] [required] Endpoint - Google Cloud Storage service endpoint
  signatureVersion: # [string] Signature version - Signature version to use for sig
  objectACL: # [string] Object ACL - Object ACL to assign to uploaded objects
  storageClass: # [string] Storage class - Storage class to select for uploaded obj
  reuseConnections: # [boolean] Reuse connections - Reuse connections between reque
  rejectUnauthorized: # [boolean] Reject unauthorized certificates - Reject certifi
  addIdToStagePath: # [boolean] Add output ID - Add the Output ID value to staging
  removeEmptyDirs: # [boolean] Remove empty staging directories - Remove empty stag

  # -------------- if removeEmptyDirs is true ---------------
```

```
  emptyDirCleanupSec: # [number; minimum: 10, maximum: 86400] Staging cleanup perio

  # ------------------------------------------------------

  maxFileOpenTimeSec: # [number; minimum: 10, maximum: 1800] File open time limit (
  maxFileIdleTimeSec: # [number; minimum: 5, maximum: 1800] Idle time limit (sec) -
  maxOpenFiles: # [number; minimum: 10, maximum: 2000] Open file limit - Maximum nu
  onBackpressure: # [string] Backpressure behavior - How to handle events when all
  deadletterEnabled: # [boolean] Enable dead-lettering - If a file fails to move to

  # -------------- if deadletterEnabled is true ---------------

  deadletterPath: # [string] Dead-letter location - Storage location for files that
  maxRetryNum: # [number; minimum: 1] Retry limit - The maximum number of times a f

  # ------------------------------------------------------

  onDiskFullBackpressure: # [string] Disk space protection - How to handle events w
  type: # [string] Output Type
  maxFileSizeMB: # [number; minimum: 5, maximum: 1024] File size limit (MB) - Maxim
  encodedConfiguration: # [string] Exabeam connection string - Enter an encoded str
  collectorInstanceId: # [string] [required] Collector instance ID - ID of the Exab

  siteName: # [string] Site name - Constant or JavaScript expression to create an E
  siteId: # [string] Site ID - Exabeam site ID. If left blank, Cribl Edge will use
  timezoneOffset: # [string] Timezone offset
  awsApiKey: # [string] Access key - HMAC access key. Can be a constant or a JavaSc
  awsSecretKey: # [string] Secret - HMAC secret. Can be a constant or a JavaScript
  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
kafka_output: # [object]
  type: # [string] Output Type
  brokers: # [array of strings] Bootstrap servers - Enter each Kafka bootstrap serv
  topic: # [string] [required] Topic - The topic to publish events to. Can be overr
  ack: # [integer] Acknowledgments - Control the number of required acknowledgments
  format: # [string] Record data format - Format to use to serialize events before

  # -------------- if format is protobuf ---------------

  protobufLibraryId: # [string] Definition set - Select a set of Protobuf definitio
```

```
# -----------------------------------------------------

compression: # [string] Compression - Codec to use to compress the data before se
maxRecordSizeKB: # [number; minimum: 1] Record size limit (KB, uncompressed) - Ma
flushEventCount: # [number; minimum: 1, maximum: 10000] Events-per-batch limit -
flushPeriodSec: # [number] Flush period (sec) - The maximum amount of time you wa
kafkaSchemaRegistry: # [object] Kafka Schema Registry Authentication
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  schemaRegistryURL: # [string] Schema Registry URL - URL for accessing the Confl
  schemaType: # [string] Schema type - The schema format used to encode and decod
  connectionTimeout: # [number; minimum: 1000, maximum: 60000] Connection timeout
  requestTimeout: # [number; minimum: 1000, maximum: 60000] Request timeout (ms)
  maxRetries: # [number; maximum: 100] Retry limit - Maximum number of times to t
  auth: # [object]  - Credentials to use when authenticating with the schema regi
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    credentialsSecret: # [string] Credentials secret - Select or create a secret

    # -------------------------------------------------------

  tls: # [object] TLS settings (client side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    rejectUnauthorized: # [boolean] Validate server certs - Reject certificates t
                trusted CA (such as the system's). Defaults to Enabled. Overrides
    servername: # [string] Server name (SNI) - Server name for the SNI (Server Na
    certificateName: # [string] Certificate - The name of the predefined certific
    caPath: # [string] CA certificate path - Path on client in which to find CA c
    privKeyPath: # [string] Private key path (mutual auth) - Path on client in wh
    certPath: # [string] Certificate path (mutual auth) - Path on client in which
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
    minVersion: # [string] Minimum TLS version
    maxVersion: # [string] Maximum TLS version

    # -------------------------------------------------------
```

```
  defaultKeySchemaId: # [number] Default key schema ID - Used when __keySchemaIdC
  defaultValueSchemaId: # [number] Default value schema ID - Used when __valueSch

  # ------------------------------------------------------

connectionTimeout: # [number; minimum: 1000, maximum: 3600000] Connection timeout
requestTimeout: # [number; minimum: 1000, maximum: 3600000] Request timeout (ms)
maxRetries: # [number; maximum: 100] Retry limit - If messages are failing, you c
maxBackOff: # [number; minimum: 30000, maximum: 180000] Backoff limit (ms) - The
initialBackoff: # [number; minimum: 300, maximum: 600000] Initial retry interval
backoffRate: # [number; minimum: 2, maximum: 20] Backoff multiplier - Set the bac
authenticationTimeout: # [number; minimum: 1000, maximum: 3600000] Authentication
reauthenticationThreshold: # [number; minimum: 1000, maximum: 1800000] Reauthenti
sasl: # [object] Authentication - Authentication parameters to use when connectir
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  mechanism: # [string] SASL mechanism
  oauthEnabled: # [boolean] Enable OAuth - Enable OAuth authentication

  # ------------------------------------------------------

tls: # [object] TLS settings (client side)
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  rejectUnauthorized: # [boolean] Validate server certs - Reject certificates tha
                trusted CA (such as the system's). Defaults to Enabled. Overrides
  servername: # [string] Server name (SNI) - Server name for the SNI (Server Name
  certificateName: # [string] Certificate - The name of the predefined certificat
  caPath: # [string] CA certificate path - Path on client in which to find CA cer
  privKeyPath: # [string] Private key path (mutual auth) - Path on client in whic
  certPath: # [string] Certificate path (mutual auth) - Path on client in which t
  passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
  minVersion: # [string] Minimum TLS version
  maxVersion: # [string] Maximum TLS version

  # ------------------------------------------------------

onBackpressure: # [string] Backpressure behavior - How to handle events when all
```

```
  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] File size limit - The maximum size to store in each que
  pqMaxSize: # [string] Queue size limit - The maximum disk space that the queue ca
  pqPath: # [string] Queue file path - The location for the persistent queue files.
  pqCompress: # [string] Compression - Codec to use to compress the persisted data
  pqOnBackpressure: # [string] Queue-full behavior - How to handle events when the
  pqMode: # [string] Mode - In Error mode, PQ writes events to the filesystem if th

  # ---------------------------------------------------------

  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
confluent_cloud_output: # [object]
  type: # [string] Output Type
  brokers: # [array of strings] Bootstrap servers - List of Confluent Cloud bootstr
  tls: # [object] TLS settings (client side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    rejectUnauthorized: # [boolean] Validate server certs - Reject certificates tha
                  trusted CA (such as the system's). Defaults to Enabled. Overrides
    servername: # [string] Server name (SNI) - Server name for the SNI (Server Name
    certificateName: # [string] Certificate - The name of the predefined certificat
    caPath: # [string] CA certificate path - Path on client in which to find CA cer
    privKeyPath: # [string] Private key path (mutual auth) - Path on client in whic
    certPath: # [string] Certificate path (mutual auth) - Path on client in which t
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
    minVersion: # [string] Minimum TLS version
    maxVersion: # [string] Maximum TLS version

    # ---------------------------------------------------------

  topic: # [string] [required] Topic - The topic to publish events to. Can be overr
  ack: # [integer] Acknowledgments - Control the number of required acknowledgments
  format: # [string] Record data format - Format to use to serialize events before

  # -------------- if format is protobuf ---------------
```

```
protobufLibraryId: # [string] Definition set - Select a set of Protobuf definitic

# ------------------------------------------------------

compression: # [string] Compression - Codec to use to compress the data before se
maxRecordSizeKB: # [number; minimum: 1] Record size limit (KB, uncompressed) - Ma
flushEventCount: # [number; minimum: 1, maximum: 10000] Events-per-batch limit -
flushPeriodSec: # [number] Flush period (sec) - The maximum amount of time you wa
kafkaSchemaRegistry: # [object] Kafka Schema Registry Authentication
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  schemaRegistryURL: # [string] Schema Registry URL - URL for accessing the Confl
  schemaType: # [string] Schema type - The schema format used to encode and decoc
  connectionTimeout: # [number; minimum: 1000, maximum: 60000] Connection timeout
  requestTimeout: # [number; minimum: 1000, maximum: 60000] Request timeout (ms)
  maxRetries: # [number; maximum: 100] Retry limit - Maximum number of times to t
  auth: # [object]  - Credentials to use when authenticating with the schema regi
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    credentialsSecret: # [string] Credentials secret - Select or create a secret

    # ------------------------------------------------------

  tls: # [object] TLS settings (client side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    rejectUnauthorized: # [boolean] Validate server certs - Reject certificates t
                trusted CA (such as the system's). Defaults to Enabled. Overrides
    servername: # [string] Server name (SNI) - Server name for the SNI (Server Na
    certificateName: # [string] Certificate - The name of the predefined certific
    caPath: # [string] CA certificate path - Path on client in which to find CA c
    privKeyPath: # [string] Private key path (mutual auth) - Path on client in wh
    certPath: # [string] Certificate path (mutual auth) - Path on client in which
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
    minVersion: # [string] Minimum TLS version
    maxVersion: # [string] Maximum TLS version
```

```
    # ------------------------------------------------------

  defaultKeySchemaId: # [number] Default key schema ID - Used when __keySchemaIdC
  defaultValueSchemaId: # [number] Default value schema ID - Used when __valueSch

  # ------------------------------------------------------

connectionTimeout: # [number; minimum: 1000, maximum: 3600000] Connection timeout
requestTimeout: # [number; minimum: 1000, maximum: 3600000] Request timeout (ms)
maxRetries: # [number; maximum: 100] Retry limit - If messages are failing, you c
maxBackOff: # [number; minimum: 30000, maximum: 180000] Backoff limit (ms) - The
initialBackoff: # [number; minimum: 300, maximum: 600000] Initial retry interval
backoffRate: # [number; minimum: 2, maximum: 20] Backoff multiplier - Set the bac
authenticationTimeout: # [number; minimum: 1000, maximum: 3600000] Authentication
reauthenticationThreshold: # [number; minimum: 1000, maximum: 1800000] Reauthenti
sasl: # [object] Authentication - Authentication parameters to use when connectir
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  mechanism: # [string] SASL mechanism
  oauthEnabled: # [boolean] Enable OAuth - Enable OAuth authentication

  # ------------------------------------------------------

onBackpressure: # [string] Backpressure behavior - How to handle events when all

# -------------- if onBackpressure is queue ---------------

pqMaxFileSize: # [string] File size limit - The maximum size to store in each que
pqMaxSize: # [string] Queue size limit - The maximum disk space that the queue ca
pqPath: # [string] Queue file path - The location for the persistent queue files.
pqCompress: # [string] Compression - Codec to use to compress the persisted data
pqOnBackpressure: # [string] Queue-full behavior - How to handle events when the
pqMode: # [string] Mode - In Error mode, PQ writes events to the filesystem if th

# -------------------------------------------------------

description: # [string] Description
pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
systemFields: # [array of strings] System fields - Fields to automatically add to
environment: # [string] Environment - Optionally, enable this config only on a sp
```

```
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
msk_output: # [object]
  type: # [string] Output Type
  brokers: # [array of strings] Bootstrap servers - Enter each Kafka bootstrap serv
  topic: # [string] [required] Topic - The topic to publish events to. Can be overr
  ack: # [integer] Acknowledgments - Control the number of required acknowledgments
  format: # [string] Record data format - Format to use to serialize events before

  # -------------- if format is protobuf ---------------

  protobufLibraryId: # [string] Definition set - Select a set of Protobuf definitio

  # ------------------------------------------------------

  compression: # [string] Compression - Codec to use to compress the data before se
  maxRecordSizeKB: # [number; minimum: 1] Record size limit (KB, uncompressed) - Ma
  flushEventCount: # [number; minimum: 1, maximum: 10000] Events-per-batch limit -
  flushPeriodSec: # [number] Flush period (sec) - The maximum amount of time you wa
  kafkaSchemaRegistry: # [object] Kafka Schema Registry Authentication
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    schemaRegistryURL: # [string] Schema Registry URL - URL for accessing the Confl
    schemaType: # [string] Schema type - The schema format used to encode and decod
    connectionTimeout: # [number; minimum: 1000, maximum: 60000] Connection timeout
    requestTimeout: # [number; minimum: 1000, maximum: 60000] Request timeout (ms)
    maxRetries: # [number; maximum: 100] Retry limit - Maximum number of times to t
    auth: # [object]  - Credentials to use when authenticating with the schema regi
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      credentialsSecret: # [string] Credentials secret - Select or create a secret

      # ------------------------------------------------------

    tls: # [object] TLS settings (client side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      rejectUnauthorized: # [boolean] Validate server certs - Reject certificates t
```

```
                trusted CA (such as the system's). Defaults to Enabled. Overrides
    servername: # [string] Server name (SNI) - Server name for the SNI (Server Na
    certificateName: # [string] Certificate - The name of the predefined certific
    caPath: # [string] CA certificate path - Path on client in which to find CA c
    privKeyPath: # [string] Private key path (mutual auth) - Path on client in wh
    certPath: # [string] Certificate path (mutual auth) - Path on client in which
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
    minVersion: # [string] Minimum TLS version
    maxVersion: # [string] Maximum TLS version

    # ------------------------------------------------------

  defaultKeySchemaId: # [number] Default key schema ID - Used when __keySchemaIdC
  defaultValueSchemaId: # [number] Default value schema ID - Used when __valueSch

  # ------------------------------------------------------

connectionTimeout: # [number; minimum: 1000, maximum: 3600000] Connection timeout
requestTimeout: # [number; minimum: 1000, maximum: 3600000] Request timeout (ms)
maxRetries: # [number; maximum: 100] Retry limit - If messages are failing, you c
maxBackOff: # [number; minimum: 30000, maximum: 180000] Backoff limit (ms) - The
initialBackoff: # [number; minimum: 300, maximum: 600000] Initial retry interval
backoffRate: # [number; minimum: 2, maximum: 20] Backoff multiplier - Set the bac
authenticationTimeout: # [number; minimum: 1000, maximum: 3600000] Authentication
reauthenticationThreshold: # [number; minimum: 1000, maximum: 1800000] Reauthenti
awsAuthenticationMethod: # [string] [required] Authentication method - AWS auther

# -------------- if awsAuthenticationMethod is manual ---------------

awsApiKey: # [string] Access key

# ------------------------------------------------------


# -------------- if awsAuthenticationMethod is secret ---------------

awsSecret: # [string] Secret key pair - Select or create a stored secret that ref

# ------------------------------------------------------

awsSecretKey: # [string] Secret key
region: # [string] [required] Region - Region where the MSK cluster is located
endpoint: # [string] Endpoint - MSK cluster service endpoint. If empty, defaults
```

```
signatureVersion: # [string] Signature version - Signature version to use for sig
reuseConnections: # [boolean] Reuse connections - Reuse connections between reque
rejectUnauthorized: # [boolean] Reject unauthorized certificates - Reject certifi
enableAssumeRole: # [boolean] Enable for MSK - Use Assume Role credentials to acc
assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the role
assumeRoleExternalId: # [string] External ID - External ID to use when assuming r
durationSeconds: # [number; minimum: 900, maximum: 43200] Duration (seconds) - Du
tls: # [object] TLS settings (client side)
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  rejectUnauthorized: # [boolean] Validate server certs - Reject certificates tha
                trusted CA (such as the system's). Defaults to Enabled. Overrides
  servername: # [string] Server name (SNI) - Server name for the SNI (Server Name
  certificateName: # [string] Certificate - The name of the predefined certificat
  caPath: # [string] CA certificate path - Path on client in which to find CA cer
  privKeyPath: # [string] Private key path (mutual auth) - Path on client in whic
  certPath: # [string] Certificate path (mutual auth) - Path on client in which t
  passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
  minVersion: # [string] Minimum TLS version
  maxVersion: # [string] Maximum TLS version

  # ---------------------------------------------------------

onBackpressure: # [string] Backpressure behavior - How to handle events when all

# -------------- if onBackpressure is queue ---------------

pqMaxFileSize: # [string] File size limit - The maximum size to store in each que
pqMaxSize: # [string] Queue size limit - The maximum disk space that the queue ca
pqPath: # [string] Queue file path - The location for the persistent queue files.
pqCompress: # [string] Compression - Codec to use to compress the persisted data
pqOnBackpressure: # [string] Queue-full behavior - How to handle events when the
pqMode: # [string] Mode - In Error mode, PQ writes events to the filesystem if th

# -----------------------------------------------------------

description: # [string] Description
pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
systemFields: # [array of strings] System fields - Fields to automatically add to
environment: # [string] Environment - Optionally, enable this config only on a sp
streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
```

```
elastic_output: # [object]
  type: # [string] Output Type
  loadBalanced: # [boolean] Load balancing - Enable for optimal performance. Even i

  # -------------- if loadBalanced is false ---------------

  url: # [string] Bulk API URL or Cloud ID - The Cloud ID or URL to an Elastic clus
  useRoundRobinDns: # [boolean] Round-robin DNS - Enable round-robin DNS lookup. Wh

  # ------------------------------------------------------



  # -------------- if loadBalanced is true ---------------

  excludeSelf: # [boolean] Exclude current host IPs - Exclude all IPs of the curren
  urls: # [array] Bulk API URLs
    - url: # [string] URL - The URL to an Elastic node to send events to. Example:
      weight: # [number] Load Weight - Assign a weight (>0) to each endpoint to inc
  dnsResolvePeriodSec: # [number; maximum: 86400] DNS resolution period (seconds) -
  loadBalanceStatsPeriodSec: # [number; minimum: 10] Load balance stats period (sec

  # ------------------------------------------------------

  index: # [string] Index or data stream - Index or data stream to send events to.
  docType: # [string] Type - Document type to use for events. Can be overwritten by
  concurrency: # [number; minimum: 1, maximum: 32] Request concurrency - Maximum nu
  maxPayloadSizeKB: # [number; minimum: 1024, maximum: 102400] Body size limit (KB)
  maxPayloadEvents: # [number] Events-per-request limit - Maximum number of events
  compress: # [boolean] Compress - Compress the payload body before sending
  rejectUnauthorized: # [boolean] Validate server certs - Reject certificates not a
      Enabled by default. When this setting is also present in TLS Settings (Client
      that value will take precedence.
  timeoutSec: # [number; minimum: 1, maximum: 9007199254740991] Request timeout - A
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
  extraHttpHeaders: # [array] Extra HTTP headers - Headers to add to all events
    - name: # [string] Field Name
      value: # [string] Field Value
  failedRequestLoggingMode: # [string] Failed request logging mode - Data to log wh
  safeHeaders: # [array of strings] Safe headers - List of headers that are safe to
  responseRetrySettings: # [array] Settings for failed HTTP requests - Automaticall
    - httpStatus: # [number; minimum: 100, maximum: 599] HTTP status code - The HTT
      initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How l
      backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base fo
```

```
    maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) -
timeoutRetrySettings: # [object]
  timeoutRetry: # [boolean] Retry timed-out HTTP requests

  # -------------- if timeoutRetry is true ---------------

  initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How lor
  backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base for
  maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) - Th

  # -------------------------------------------------------

responseHonorRetryAfterHeader: # [boolean] Honor Retry-After header - Honor any F
extraParams: # [array] Extra parameters
  - name: # [string] Field Name
    value: # [string] Field Value
auth: # [object]
  disabled: # [boolean] Authentication Disabled

  # -------------- if disabled is false ---------------

  authType: # [string] Authentication method - Enter credentials directly, or sel

  # -------------------------------------------------------

elasticVersion: # [string] Elastic version - Optional Elasticsearch version, usec
elasticPipeline: # [string] Elastic pipeline - Optional Elasticsearch destinatior
includeDocId: # [boolean] Include document _id - Include the `document_id` field
writeAction: # [string] Write action - Action to use when writing events. Must be
retryPartialErrors: # [boolean] Retry partial errors - Retry failed events when a
onBackpressure: # [string] Backpressure behavior - How to handle events when all

# -------------- if onBackpressure is queue ---------------

pqMaxFileSize: # [string] File size limit - The maximum size to store in each que
pqMaxSize: # [string] Queue size limit - The maximum disk space that the queue ca
pqPath: # [string] Queue file path - The location for the persistent queue files.
pqCompress: # [string] Compression - Codec to use to compress the persisted data
pqOnBackpressure: # [string] Queue-full behavior - How to handle events when the
pqMode: # [string] Mode - In Error mode, PQ writes events to the filesystem if th

# -------------------------------------------------------
```

```
  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
elastic_cloud_output: # [object]
  type: # [string] Output Type
  url: # [string] Cloud ID - Enter Cloud ID of the Elastic Cloud environment to ser
  index: # [string] [required] Data stream or index - Data stream or index to send
  concurrency: # [number; minimum: 1, maximum: 32] Request concurrency - Maximum nu
  maxPayloadSizeKB: # [number; minimum: 1024, maximum: 102400] Body size limit (KB)
  maxPayloadEvents: # [number] Events-per-request limit - Maximum number of events
  compress: # [boolean] Compress - Compress the payload body before sending
  rejectUnauthorized: # [boolean] Validate server certs - Reject certificates not a
      Enabled by default. When this setting is also present in TLS Settings (Client
      that value will take precedence.
  timeoutSec: # [number; minimum: 1, maximum: 9007199254740991] Request timeout - A
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
  extraHttpHeaders: # [array] Extra HTTP headers - Headers to add to all events
    - name: # [string] Field Name
      value: # [string] Field Value
  failedRequestLoggingMode: # [string] Failed request logging mode - Data to log wh
  safeHeaders: # [array of strings] Safe headers - List of headers that are safe to
  extraParams: # [array] Extra parameters - Extra parameters to use in HTTP request
    - name: # [string] Field Name
      value: # [string] Field Value
  auth: # [object]
    disabled: # [boolean] Authentication Disabled

    # -------------- if disabled is false ---------------

    authType: # [string] Authentication method - Enter credentials directly, or sel

    # ----------------------------------------------------

  elasticPipeline: # [string] Elastic pipeline - Optional Elastic Cloud Destinatior
  includeDocId: # [boolean] Include document _id - Include the `document_id` field
  responseRetrySettings: # [array] Settings for failed HTTP requests - Automatical
    - httpStatus: # [number; minimum: 100, maximum: 599] HTTP status code - The HTT
      initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How l
      backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base fo
      maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) -
  timeoutRetrySettings: # [object]
```

```
    timeoutRetry: # [boolean] Retry timed-out HTTP requests

    # -------------- if timeoutRetry is true ---------------

    initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How lor
    backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base for
    maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) - Th

    # -------------------------------------------------------

  responseHonorRetryAfterHeader: # [boolean] Honor Retry-After header - Honor any R
  onBackpressure: # [string] Backpressure behavior - How to handle events when all

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] File size limit - The maximum size to store in each que
  pqMaxSize: # [string] Queue size limit - The maximum disk space that the queue ca
  pqPath: # [string] Queue file path - The location for the persistent queue files.
  pqCompress: # [string] Compression - Codec to use to compress the persisted data
  pqOnBackpressure: # [string] Queue-full behavior - How to handle events when the
  pqMode: # [string] Mode - In Error mode, PQ writes events to the filesystem if th

  # -------------------------------------------------------

  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
newrelic_output: # [object]
  type: # [string] Output Type
  region: # [string] Region - Which New Relic region endpoint to use.

  # -------------- if region is Custom ---------------

  customUrl: # [string]

  # -------------------------------------------------------

  logType: # [string] Log type - Name of the logtype to send with events, e.g.: obs
  messageField: # [string] Log message field - Name of field to send as log message
  metadata: # [array] Fields - Fields to add to events from this input
    - name: # [string] Field Name
```

```
    value: # [string] Value - JavaScript expression to compute field's value, enc
concurrency: # [number; minimum: 1, maximum: 32] Request concurrency - Maximum nu
maxPayloadSizeKB: # [number; minimum: 1, maximum: 1024] Body size limit (KB) - Ma
maxPayloadEvents: # [number] Events-per-request limit - Maximum number of events
compress: # [boolean] Compress - Compress the payload body before sending
rejectUnauthorized: # [boolean] Validate server certs - Reject certificates not a
    Enabled by default. When this setting is also present in TLS Settings (Client
    that value will take precedence.
timeoutSec: # [number; minimum: 1, maximum: 9007199254740991] Request timeout - A
flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
extraHttpHeaders: # [array] Extra HTTP headers - Headers to add to all events
  - name: # [string] Field Name
    value: # [string] Field Value
useRoundRobinDns: # [boolean] Round-robin DNS - Enable round-robin DNS lookup. Wh
failedRequestLoggingMode: # [string] Failed request logging mode - Data to log wh
safeHeaders: # [array of strings] Safe headers - List of headers that are safe to
responseRetrySettings: # [array] Settings for failed HTTP requests - Automaticall
  - httpStatus: # [number; minimum: 100, maximum: 599] HTTP status code - The HTT
    initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How l
    backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base fo
    maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) -
timeoutRetrySettings: # [object]
  timeoutRetry: # [boolean] Retry timed-out HTTP requests

  # -------------- if timeoutRetry is true ---------------

  initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How lon
  backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base for
  maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) - Th

  # -------------------------------------------------------

responseHonorRetryAfterHeader: # [boolean] Honor Retry-After header - Honor any R
onBackpressure: # [string] Backpressure behavior - How to handle events when all

# -------------- if onBackpressure is queue ---------------

pqMaxFileSize: # [string] File size limit - The maximum size to store in each que
pqMaxSize: # [string] Queue size limit - The maximum disk space that the queue ca
pqPath: # [string] Queue file path - The location for the persistent queue files.
pqCompress: # [string] Compression - Codec to use to compress the persisted data
pqOnBackpressure: # [string] Queue-full behavior - How to handle events when the
pqMode: # [string] Mode - In Error mode, PQ writes events to the filesystem if th
```

```
  # ------------------------------------------------------

  authType: # [string] Authentication method - Enter API key directly, or select a
  apiKey: # [string] API key - New Relic API key. Can be overridden using __newReli

  # -------------- if authType is manual ---------------


  # ------------------------------------------------------

  textSecret: # [string] API key (text secret) - Select or create a stored text sec

  # -------------- if authType is secret ---------------


  # ------------------------------------------------------

  totalMemoryLimitKB: # [number] Buffer memory limit (KB) - Maximum total size of t
  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
newrelic_events_output: # [object]
  type: # [string] Output Type
  region: # [string] Region - Which New Relic region endpoint to use.

  # -------------- if region is Custom ---------------

  customUrl: # [string]

  # ------------------------------------------------------

  accountId: # [string] Account ID - New Relic account ID
  eventType: # [string] [required] Event type - Default eventType to use when not p
  concurrency: # [number; minimum: 1, maximum: 32] Request concurrency - Maximum nu
  maxPayloadSizeKB: # [number; minimum: 1, maximum: 1024] Body size limit (KB) - Ma
  maxPayloadEvents: # [number] Events-per-request limit - Maximum number of events
  compress: # [boolean] Compress - Compress the payload body before sending
  rejectUnauthorized: # [boolean] Validate server certs - Reject certificates not a
      Enabled by default. When this setting is also present in TLS Settings (Client
      that value will take precedence.
```

```
timeoutSec: # [number; minimum: 1, maximum: 9007199254740991] Request timeout - A
flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
extraHttpHeaders: # [array] Extra HTTP headers - Headers to add to all events
  - name: # [string] Field Name
    value: # [string] Field Value
useRoundRobinDns: # [boolean] Round-robin DNS - Enable round-robin DNS lookup. Wh
failedRequestLoggingMode: # [string] Failed request logging mode - Data to log wh
safeHeaders: # [array of strings] Safe headers - List of headers that are safe to
responseRetrySettings: # [array] Settings for failed HTTP requests - Automaticall
  - httpStatus: # [number; minimum: 100, maximum: 599] HTTP status code - The HTT
    initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How l
    backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base fo
    maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) -
timeoutRetrySettings: # [object]
  timeoutRetry: # [boolean] Retry timed-out HTTP requests

  # -------------- if timeoutRetry is true ---------------

  initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How lon
  backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base for
  maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) - Th

  # ------------------------------------------------------

responseHonorRetryAfterHeader: # [boolean] Honor Retry-After header - Honor any R
onBackpressure: # [string] Backpressure behavior - How to handle events when all

# -------------- if onBackpressure is queue ---------------

pqMaxFileSize: # [string] File size limit - The maximum size to store in each que
pqMaxSize: # [string] Queue size limit - The maximum disk space that the queue ca
pqPath: # [string] Queue file path - The location for the persistent queue files.
pqCompress: # [string] Compression - Codec to use to compress the persisted data
pqOnBackpressure: # [string] Queue-full behavior - How to handle events when the
pqMode: # [string] Mode - In Error mode, PQ writes events to the filesystem if th

# ------------------------------------------------------

authType: # [string] Authentication method - Enter API key directly, or select a
apiKey: # [string] API key - New Relic API key. Can be overridden using __newReli

# -------------- if authType is manual ---------------
```

```
  # ------------------------------------------------------

  textSecret: # [string] API key (text secret) - Select or create a stored text sec

  # -------------- if authType is secret ---------------


  # ------------------------------------------------------

  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
influxdb_output: # [object]
  type: # [string] Output Type
  url: # [string] Write API URL - URL of an InfluxDB cluster to send events to, e.g
  useV2API: # [boolean] Use v2 API - The v2 API can be enabled with InfluxDB versio

  # -------------- if useV2API is false ---------------

  database: # [string] Database - Database to write to.

  # ------------------------------------------------------


  # -------------- if useV2API is true ---------------

  bucket: # [string] Bucket - Bucket to write to.
  org: # [string] Organization - Organization ID for this bucket.

  # ------------------------------------------------------

  timestampPrecision: # [string] Timestamp precision - Sets the precision for the s
  dynamicValueFieldName: # [boolean] Dynamic value fields - Enabling this will pull
  valueFieldName: # [string] Value field name - Name of the field in which to store
  concurrency: # [number; minimum: 1, maximum: 32] Request concurrency - Maximum nu
  maxPayloadSizeKB: # [number; minimum: 1024, maximum: 51200] Body size limit (KB)
  maxPayloadEvents: # [number] Events-per-request limit - Maximum number of events
  compress: # [boolean] Compress - Compress the payload body before sending
  rejectUnauthorized: # [boolean] Validate server certs - Reject certificates not a
      Enabled by default. When this setting is also present in TLS Settings (Client
```

```
    that value will take precedence.
timeoutSec: # [number; minimum: 1, maximum: 9007199254740991] Request timeout - A
flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
extraHttpHeaders: # [array] Extra HTTP headers - Headers to add to all events
  - name: # [string] Field Name
    value: # [string] Field Value
useRoundRobinDns: # [boolean] Round-robin DNS - Enable round-robin DNS lookup. Wh
failedRequestLoggingMode: # [string] Failed request logging mode - Data to log wh
safeHeaders: # [array of strings] Safe headers - List of headers that are safe to
responseRetrySettings: # [array] Settings for failed HTTP requests - Automaticall
  - httpStatus: # [number; minimum: 100, maximum: 599] HTTP status code - The HTT
    initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How l
    backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base fo
    maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) -
timeoutRetrySettings: # [object]
  timeoutRetry: # [boolean] Retry timed-out HTTP requests

  # -------------- if timeoutRetry is true ---------------

  initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How lon
  backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base for
  maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) - Th

  # -------------------------------------------------------

responseHonorRetryAfterHeader: # [boolean] Honor Retry-After header - Honor any F
onBackpressure: # [string] Backpressure behavior - How to handle events when all

# -------------- if onBackpressure is queue ---------------

pqMaxFileSize: # [string] File size limit - The maximum size to store in each que
pqMaxSize: # [string] Queue size limit - The maximum disk space that the queue ca
pqPath: # [string] Queue file path - The location for the persistent queue files.
pqCompress: # [string] Compression - Codec to use to compress the persisted data
pqOnBackpressure: # [string] Queue-full behavior - How to handle events when the
pqMode: # [string] Mode - In Error mode, PQ writes events to the filesystem if th

# -------------------------------------------------------

authType: # [string] Authentication type - InfluxDB authentication type

# -------------- if authType is basic ---------------
```

```
username: # [string] Username
password: # [string] Password

# ------------------------------------------------------


# -------------- if authType is token ---------------

token: # [string] Token - Bearer token to include in the authorization header

# ------------------------------------------------------


# -------------- if authType is credentialsSecret ---------------

credentialsSecret: # [string] Credentials secret - Select or create a secret that

# ------------------------------------------------------


# -------------- if authType is textSecret ---------------

textSecret: # [string] Token (text secret) - Select or create a stored text secre

# ------------------------------------------------------


# -------------- if authType is oauth ---------------

loginUrl: # [string] Login URL - URL for OAuth
secretParamName: # [string] OAuth Secret parameter name - Secret parameter name t
secret: # [string] OAuth secret - Secret parameter value to pass in request body
tokenAttributeName: # [string] Token attribute name - Name of the auth token attr
spacer: # [null]
authHeaderExpr: # [string] Authorize expression - JavaScript expression to comput
tokenTimeoutSecs: # [number; minimum: 1, maximum: 300000] Refresh interval (secs.
oauthParams: # [array] OAuth parameters - Additional parameters to send in the OA
  - name: # [string] Name - OAuth parameter name
    value: # [string] Value - OAuth parameter value
oauthHeaders: # [array] OAuth headers - Additional headers to send in the OAuth l
  - name: # [string] Name - OAuth header name
    value: # [string] Value - OAuth header value
```

```
  # ------------------------------------------------------

  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
cloudwatch_output: # [object]
  type: # [string] Output Type
  logGroupName: # [string] Log group name - CloudWatch log group to associate event
  logStreamName: # [string] [required] Log stream prefix - Prefix for CloudWatch lo
  awsAuthenticationMethod: # [string] Authentication method - AWS authentication me

  # -------------- if awsAuthenticationMethod is manual ---------------

  awsApiKey: # [string] Access key

  # ------------------------------------------------------


  # -------------- if awsAuthenticationMethod is secret ---------------

  awsSecret: # [string] Secret key pair - Select or create a stored secret that ref

  # ------------------------------------------------------

  awsSecretKey: # [string] Secret key
  region: # [string] [required] Region - Region where the CloudWatchLogs is located
  endpoint: # [string] Endpoint - CloudWatchLogs service endpoint. If empty, defaul
  reuseConnections: # [boolean] Reuse connections - Reuse connections between reque
  rejectUnauthorized: # [boolean] Reject unauthorized certificates - Reject certifi
  enableAssumeRole: # [boolean] Enable for CloudWatchLogs - Use Assume Role credent
  assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the role
  assumeRoleExternalId: # [string] External ID - External ID to use when assuming r
  durationSeconds: # [number; minimum: 900, maximum: 43200] Duration (seconds) - Du
  maxQueueSize: # [number; minimum: 1, maximum: 32] Queue size limit - Maximum numb
  maxRecordSizeKB: # [number; minimum: 1, maximum: 10240] Record size limit (KB, ur
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
  onBackpressure: # [string] Backpressure behavior - How to handle events when all

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] File size limit - The maximum size to store in each que
```

```
  pqMaxSize: # [string] Queue size limit - The maximum disk space that the queue ca
  pqPath: # [string] Queue file path - The location for the persistent queue files.
  pqCompress: # [string] Compression - Codec to use to compress the persisted data
  pqOnBackpressure: # [string] Queue-full behavior - How to handle events when the
  pqMode: # [string] Mode - In Error mode, PQ writes events to the filesystem if th

  # ------------------------------------------------------

  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
minio_output: # [object]
  type: # [string] Output Type
  endpoint: # [string] [required] MinIO endpoint - MinIO service url (e.g. http://m
  bucket: # [string] MinIO bucket name - Name of the destination MinIO bucket. This
  awsAuthenticationMethod: # [string] Authentication method - AWS authentication me

  # -------------- if awsAuthenticationMethod is manual ---------------

  awsApiKey: # [string] Access key - This value can be a constant or a JavaScript e

  # ------------------------------------------------------


  # -------------- if awsAuthenticationMethod is secret ---------------

  awsSecret: # [string] Secret key pair - Select or create a stored secret that ref

  # ------------------------------------------------------

  awsSecretKey: # [string] Secret key - Secret key. This value can be a constant or
  region: # [string] Region - Region where the MinIO service/cluster is located
  stagePath: # [string] [required] Staging location - Filesystem location in which
  addIdToStagePath: # [boolean] Add output ID - Add the Output ID value to staging
  destPath: # [string] Key prefix - Root directory to prepend to path before upload
  signatureVersion: # [string] Signature version - Signature version to use for sig
  objectACL: # [string] Object ACL - Object ACL to assign to uploaded objects
  storageClass: # [string] Storage class - Storage class to select for uploaded obj
  serverSideEncryption: # [string] Server-side encryption - Server-side encryption
  reuseConnections: # [boolean] Reuse connections - Reuse connections between reque
  rejectUnauthorized: # [boolean] Reject unauthorized certificates - Reject certifi
```

```
verifyPermissions: # [boolean] Verify if bucket exists - Disable if you can acces
removeEmptyDirs: # [boolean] Remove empty staging directories - Remove empty stag

# -------------- if removeEmptyDirs is true ---------------

emptyDirCleanupSec: # [number; minimum: 10, maximum: 86400] Staging cleanup perio

# ---------------------------------------------------------

partitionExpr: # [string] Partitioning expression - JavaScript expression definin
format: # [string] Data format - Format of the output data

# -------------- if format is json ---------------

compress: # [string] Compression - Data compression format to apply to HTTP conte
compressionLevel: # [string] Compression level - Compression level to apply befor

# ---------------------------------------------------------


# -------------- if format is parquet ---------------

automaticSchema: # [boolean] Automatic schema - Automatically calculate the schem
parquetVersion: # [string] Parquet version - Determines which data types are supp
parquetDataPageVersion: # [string] Data page version - Serialization format of da
parquetRowGroupLength: # [number; minimum: 1, maximum: 67108864] Group row limit
parquetPageSize: # [string] Page size - Target memory size for page segments, suc
shouldLogInvalidRows: # [boolean] Log invalid rows - Log up to 3 rows that Cribl
keyValueMetadata: # [array] Metadata (optional) - The metadata of files the Desti
  - key: # [string] Key
    value: # [string] Value
enableStatistics: # [boolean] Write statistics - Statistics profile an entire fil
enableWritePageIndex: # [boolean] Write page indexes - One page index contains st
enablePageChecksum: # [boolean] Write page checksum - Parquet tools can use the c

# ---------------------------------------------------------

baseFileName: # [string] File name prefix expression - JavaScript expression to c
fileNameSuffix: # [string] File name suffix expression - JavaScript expression to
maxFileSizeMB: # [number; minimum: 5, maximum: 1024] File size limit (MB) - Maxim
maxOpenFiles: # [number; minimum: 10, maximum: 2000] Open file limit - Maximum nu
headerLine: # [string] Header line - If set, this line will be written to the beg
writeHighWaterMark: # [number; minimum: 16, maximum: 4096] Writing high watermark
```

```
  onBackpressure: # [string] Backpressure behavior - How to handle events when all
  deadletterEnabled: # [boolean] Enable dead-lettering - If a file fails to move to

  # -------------- if deadletterEnabled is true ---------------

  deadletterPath: # [string] Dead-letter location - Storage location for files that
  maxRetryNum: # [number; minimum: 1] Retry limit - The maximum number of times a f

  # ------------------------------------------------------

  onDiskFullBackpressure: # [string] Disk space protection - How to handle events w
  maxFileOpenTimeSec: # [number; minimum: 10, maximum: 86400] File open time limit
  maxFileIdleTimeSec: # [number; minimum: 5, maximum: 86400] Idle time limit (sec)
  maxConcurrentFileParts: # [number; minimum: 1, maximum: 10] Concurrent file parts
  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
statsd_output: # [object]
  type: # [string] Output Type
  protocol: # [string] Destination protocol - Protocol to use when communicating wi

  # -------------- if protocol is udp ---------------

  dnsResolvePeriodSec: # [number; maximum: 86400] DNS resolution period (sec) - How

  # ------------------------------------------------------



  # -------------- if protocol is tcp ---------------

  throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to throttl
  connectionTimeout: # [number] Connection timeout - Amount of time (milliseconds)
  writeTimeout: # [number] Write timeout - Amount of time (milliseconds) to wait fo
  onBackpressure: # [string] Backpressure behavior - How to handle events when all

  # ------------------------------------------------------

  host: # [string] [required] Host - The hostname of the destination.
  port: # [number; minimum: 1, maximum: 65535] [required] Port - Destination port.
  mtu: # [number; minimum: 1, maximum: 65535] Record size limit (bytes) - When prot
  flushPeriodSec: # [number] Flush period (sec) - When protocol is TCP, specifies h
```

```
  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
statsd_ext_output: # [object]
  type: # [string] Output Type
  protocol: # [string] Destination protocol - Protocol to use when communicating wi

  # -------------- if protocol is udp ---------------

  dnsResolvePeriodSec: # [number; maximum: 86400] DNS resolution period (sec) - How

  # -------------------------------------------------------


  # -------------- if protocol is tcp ---------------

  throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to throttl
  connectionTimeout: # [number] Connection timeout - Amount of time (milliseconds)
  writeTimeout: # [number] Write timeout - Amount of time (milliseconds) to wait fo
  onBackpressure: # [string] Backpressure behavior - How to handle events when all

  # -------------------------------------------------------

  host: # [string] [required] Host - The hostname of the destination.
  port: # [number; minimum: 1, maximum: 65535] [required] Port - Destination port.
  mtu: # [number; minimum: 1, maximum: 65535] Record size limit (bytes) - When prot
  flushPeriodSec: # [number] Flush period (sec) - When protocol is TCP, specifies h
  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
graphite_output: # [object]
  type: # [string] Output Type
  protocol: # [string] Destination protocol - Protocol to use when communicating wi

  # -------------- if protocol is udp ---------------

  dnsResolvePeriodSec: # [number; maximum: 86400] DNS resolution period (sec) - How

  # -------------------------------------------------------
```

```
  # -------------- if protocol is tcp ---------------

  throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to throttl
  connectionTimeout: # [number] Connection timeout - Amount of time (milliseconds)
  writeTimeout: # [number] Write timeout - Amount of time (milliseconds) to wait fo
  onBackpressure: # [string] Backpressure behavior - How to handle events when all

  # -------------------------------------------------------

  host: # [string] [required] Host - The hostname of the destination.
  port: # [number; minimum: 1, maximum: 65535] [required] Port - Destination port.
  mtu: # [number; minimum: 1, maximum: 65535] Record size limit (bytes) - When prot
  flushPeriodSec: # [number] Flush period (sec) - When protocol is TCP, specifies h
  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
router_output: # [object]
  type: # [string] Output Type
  rules: # [array] Rules - Event routing rules
    - filter: # [string] Filter Expression - JavaScript expression to select events
      output: # [string] Output - Output to send matching events to
      description: # [string] Description - Description of this rule's purpose
      final: # [boolean] Final - Flag to control whether to stop the event from bei
  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
sns_output: # [object]
  type: # [string] Output Type
  topicArn: # [string] Topic ARN - The ARN of the SNS topic to send events to. When
  messageGroupId: # [string] [required] Message Group ID - Messages in the same gro
  maxRetries: # [number] Maximum number of retries - Maximum number of retries befo
  awsAuthenticationMethod: # [string] Authentication method - AWS authentication me

  # -------------- if awsAuthenticationMethod is manual ---------------

  awsApiKey: # [string] Access key
```

```
  # ------------------------------------------------------


  # -------------- if awsAuthenticationMethod is secret ---------------

  awsSecret: # [string] Secret key pair - Select or create a stored secret that ref

  # ------------------------------------------------------


  awsSecretKey: # [string] Secret key
  region: # [string] Region - Region where the SNS is located
  endpoint: # [string] Endpoint - SNS service endpoint. If empty, defaults to the A
  signatureVersion: # [string] Signature version - Signature version to use for sig
  reuseConnections: # [boolean] Reuse connections - Reuse connections between reque
  rejectUnauthorized: # [boolean] Reject unauthorized certificates - Reject certifi
  enableAssumeRole: # [boolean] Enable for SNS - Use Assume Role credentials to acc
  assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the role
  assumeRoleExternalId: # [string] External ID - External ID to use when assuming r
  durationSeconds: # [number; minimum: 900, maximum: 43200] Duration (seconds) - Du
  onBackpressure: # [string] Backpressure behavior - How to handle events when all

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] File size limit - The maximum size to store in each que
  pqMaxSize: # [string] Queue size limit - The maximum disk space that the queue ca
  pqPath: # [string] Queue file path - The location for the persistent queue files.
  pqCompress: # [string] Compression - Codec to use to compress the persisted data
  pqOnBackpressure: # [string] Queue-full behavior - How to handle events when the
  pqMode: # [string] Mode - In Error mode, PQ writes events to the filesystem if th

  # ------------------------------------------------------


  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
sqs_output: # [object]
  type: # [string] Output Type
  queueName: # [string] Queue Name - The name, URL, or ARN of the SQS queue to send
  queueType: # [string] [required] Queue Type - The queue type used (or created). D
  awsAccountId: # [string] AWS account ID - SQS queue owner's AWS account ID. Leave
  messageGroupId: # [string] Message Group ID - This parameter applies only to FIFO
```

```
createQueue: # [boolean] Create Queue - Create queue if it does not exist.
awsAuthenticationMethod: # [string] Authentication method - AWS authentication me

# -------------- if awsAuthenticationMethod is manual ---------------

awsApiKey: # [string] Access key

# -------------------------------------------------------


# -------------- if awsAuthenticationMethod is secret ---------------

awsSecret: # [string] Secret key pair - Select or create a stored secret that ref

# -------------------------------------------------------

awsSecretKey: # [string] Secret key
region: # [string] Region - AWS Region where the SQS queue is located. Required,
endpoint: # [string] Endpoint - SQS service endpoint. If empty, defaults to the A
signatureVersion: # [string] Signature version - Signature version to use for sig
reuseConnections: # [boolean] Reuse connections - Reuse connections between reque
rejectUnauthorized: # [boolean] Reject unauthorized certificates - Reject certifi
enableAssumeRole: # [boolean] Enable for SQS - Use Assume Role credentials to acc
assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the role
assumeRoleExternalId: # [string] External ID - External ID to use when assuming r
durationSeconds: # [number; minimum: 900, maximum: 43200] Duration (seconds) - Du
maxQueueSize: # [number; minimum: 1] Queue size limit - Maximum number of queued
maxRecordSizeKB: # [number; minimum: 1, maximum: 256] Record size limit (KB) - Ma
flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
maxInProgress: # [number; minimum: 1, maximum: 100] Concurrent request limit - Th
onBackpressure: # [string] Backpressure behavior - How to handle events when all

# -------------- if onBackpressure is queue ---------------

pqMaxFileSize: # [string] File size limit - The maximum size to store in each que
pqMaxSize: # [string] Queue size limit - The maximum disk space that the queue ca
pqPath: # [string] Queue file path - The location for the persistent queue files.
pqCompress: # [string] Compression - Codec to use to compress the persisted data
pqOnBackpressure: # [string] Queue-full behavior - How to handle events when the
pqMode: # [string] Mode - In Error mode, PQ writes events to the filesystem if th

# -------------------------------------------------------
```

```
  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
snmp_output: # [object]
  type: # [string] Output Type
  hosts: # [array] SNMP Trap Destinations - One or more SNMP destinations to forwar
    - host: # [string] Address - Destination host
      port: # [number; maximum: 65535] Port - Destination port, default is 162
  dnsResolvePeriodSec: # [number; maximum: 86400] DNS resolution period (sec) - How
  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
sumo_logic_output: # [object]
  type: # [string] Output Type
  url: # [string] API URL - Sumo Logic HTTP collector URL to which events should be
  customSource: # [string] Custom source name - Override the source name configured
  customCategory: # [string] Custom source category - Override the source category
  format: # [string] Data format - Preserve the raw event format instead of JSONify
  concurrency: # [number; minimum: 1, maximum: 32] Request concurrency - Maximum nu
  maxPayloadSizeKB: # [number; minimum: 1, maximum: 1024] Body size limit (KB) - Ma
  maxPayloadEvents: # [number] Events-per-request limit - Maximum number of events
  compress: # [boolean] Compress - Compress the payload body before sending
  rejectUnauthorized: # [boolean] Validate server certs - Reject certificates not a
      Enabled by default. When this setting is also present in TLS Settings (Client
      that value will take precedence.
  timeoutSec: # [number; minimum: 1, maximum: 9007199254740991] Request timeout - A
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
  extraHttpHeaders: # [array] Extra HTTP headers - Headers to add to all events
    - name: # [string] Field Name
      value: # [string] Field Value
  useRoundRobinDns: # [boolean] Round-robin DNS - Enable round-robin DNS lookup. Wh
  failedRequestLoggingMode: # [string] Failed request logging mode - Data to log wh
  safeHeaders: # [array of strings] Safe headers - List of headers that are safe to
  responseRetrySettings: # [array] Settings for failed HTTP requests - Automatically
    - httpStatus: # [number; minimum: 100, maximum: 599] HTTP status code - The HTT
      initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How l
      backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base fo
      maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) -
  timeoutRetrySettings: # [object]
```

```
    timeoutRetry: # [boolean] Retry timed-out HTTP requests

    # -------------- if timeoutRetry is true ---------------

    initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How lor
    backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base for
    maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) - Th

    # ------------------------------------------------------

  responseHonorRetryAfterHeader: # [boolean] Honor Retry-After header - Honor any F
  onBackpressure: # [string] Backpressure behavior - How to handle events when all

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] File size limit - The maximum size to store in each que
  pqMaxSize: # [string] Queue size limit - The maximum disk space that the queue ca
  pqPath: # [string] Queue file path - The location for the persistent queue files.
  pqCompress: # [string] Compression - Codec to use to compress the persisted data
  pqOnBackpressure: # [string] Queue-full behavior - How to handle events when the
  pqMode: # [string] Mode - In Error mode, PQ writes events to the filesystem if th

  # ------------------------------------------------------

  totalMemoryLimitKB: # [number] Buffer memory limit (KB) - Maximum total size of t
  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
datadog_output: # [object]
  type: # [string] Output Type
  contentType: # [string] Send logs as - The content type to use when sending logs
  message: # [string] Message field - Name of the event field that contains the mes
  source: # [string] Source - Name of the source to send with logs. When you send l
  host: # [string] Host - Name of the host to send with logs. When you send logs as
  service: # [string] Service - Name of the service to send with logs. When you ser
  tags: # [array of strings] Datadog tags - List of tags to send with logs, such as
  batchByTags: # [boolean] Batch by tags - Batch events by API key and the ddtags f
  allowApiKeyFromEvents: # [boolean] Allow API key from events - Allow API key to b
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
  severity: # [string] Severity - Default value for message severity. When you send
  site: # [string] Datadog site - Datadog site to which events should be sent
```

```
# -------------- if site is custom ---------------

customUrl: # [string]

# ------------------------------------------------------

sendCountersAsCount: # [boolean] Send counter metrics as 'count' - If not enablec
concurrency: # [number; minimum: 1, maximum: 32] Request concurrency - Maximum nu
maxPayloadSizeKB: # [number; minimum: 1024, maximum: 10240] Body size limit (KB)
maxPayloadEvents: # [number] Events-per-request limit - Maximum number of events
compress: # [boolean] Compress - Compress the payload body before sending
rejectUnauthorized: # [boolean] Validate server certs - Reject certificates not a
    Enabled by default. When this setting is also present in TLS Settings (Client
    that value will take precedence.
timeoutSec: # [number; minimum: 1, maximum: 9007199254740991] Request timeout - A
flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
extraHttpHeaders: # [array] Extra HTTP headers - Headers to add to all events
  - name: # [string] Field Name
    value: # [string] Field Value
useRoundRobinDns: # [boolean] Round-robin DNS - Enable round-robin DNS lookup. Wh
failedRequestLoggingMode: # [string] Failed request logging mode - Data to log wh
safeHeaders: # [array of strings] Safe headers - List of headers that are safe to
responseRetrySettings: # [array] Settings for failed HTTP requests - Automaticall
  - httpStatus: # [number; minimum: 100, maximum: 599] HTTP status code - The HTT
    initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How l
    backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base fo
    maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) -
timeoutRetrySettings: # [object]
  timeoutRetry: # [boolean] Retry timed-out HTTP requests

  # -------------- if timeoutRetry is true ---------------

  initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How lor
  backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base for
  maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) - Th

  # ------------------------------------------------------

responseHonorRetryAfterHeader: # [boolean] Honor Retry-After header - Honor any R
onBackpressure: # [string] Backpressure behavior - How to handle events when all

# -------------- if onBackpressure is queue ---------------
```

```
  pqMaxFileSize: # [string] File size limit - The maximum size to store in each que
  pqMaxSize: # [string] Queue size limit - The maximum disk space that the queue ca
  pqPath: # [string] Queue file path - The location for the persistent queue files.
  pqCompress: # [string] Compression - Codec to use to compress the persisted data
  pqOnBackpressure: # [string] Queue-full behavior - How to handle events when the
  pqMode: # [string] Mode - In Error mode, PQ writes events to the filesystem if th

  # ------------------------------------------------------

  authType: # [string] Authentication method - Enter API key directly, or select a
  apiKey: # [string] API key - Organization's API key in Datadog

  # -------------- if authType is manual ---------------


  # ------------------------------------------------------

  textSecret: # [string] API key (text secret) - Select or create a stored text sec

  # -------------- if authType is secret ---------------


  # ------------------------------------------------------

  totalMemoryLimitKB: # [number] Buffer memory limit (KB) - Maximum total size of t
  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
grafana_cloud_output: # [object]
  type: # [string] Output Type
  lokiUrl: # [string] Loki URL - The endpoint to send logs to, such as https://logs
  prometheusUrl: # [string] Prometheus URL - The remote_write endpoint to send Prom
  message: # [string] Logs message field - Name of the event field that contains th
  messageFormat: # [string] Message format - Format to use when sending logs to Lok

  # -------------- if messageFormat is json ---------------

  compress: # [boolean] Compress - Compress the payload body before sending. Applie

  # ------------------------------------------------------
```

```
labels: # [array] Logs labels - List of labels to send with logs. Labels define l
  - name: # [string] Name
    value: # [string] Value
metricRenameExpr: # [string] Metrics renaming expression - JavaScript expression
prometheusAuth: # [object]
  authType: # [string] Authentication type

  # -------------- if authType is token ---------------

  token: # [string] Auth token - Bearer token to include in the authorization hea

  # ----------------------------------------------------


  # -------------- if authType is textSecret ---------------

  textSecret: # [string] Auth token (text secret) - Select or create a stored tex

  # ----------------------------------------------------


  # -------------- if authType is basic ---------------

  username: # [string] Username - Username for authentication
  password: # [string] Password - Password (API key in Grafana Cloud domain) for

  # ----------------------------------------------------


  # -------------- if authType is credentialsSecret ---------------

  credentialsSecret: # [string] Credentials secret - Select or create a secret th

  # ----------------------------------------------------

lokiAuth: # [object]
  authType: # [string] Authentication type

  # -------------- if authType is token ---------------

  token: # [string] Auth token - Bearer token to include in the authorization hea

  # ----------------------------------------------------
```

```
  # -------------- if authType is textSecret ---------------

  textSecret: # [string] Auth token (text secret) - Select or create a stored tex

  # ---------------------------------------------------------


  # -------------- if authType is basic ---------------

  username: # [string] Username - Username for authentication
  password: # [string] Password - Password (API key in Grafana Cloud domain) for

  # ---------------------------------------------------------


  # -------------- if authType is credentialsSecret ---------------

  credentialsSecret: # [string] Credentials secret - Select or create a secret th

  # ---------------------------------------------------------

concurrency: # [number; minimum: 1, maximum: 32] Request concurrency - Maximum nu
maxPayloadSizeKB: # [number; minimum: 1024, maximum: 10240] Body size limit (KB)
maxPayloadEvents: # [number] Events-per-request limit - Maximum number of events
rejectUnauthorized: # [boolean] Validate server certs - Reject certificates not a
    Enabled by default. When this setting is also present in TLS Settings (Client
    that value will take precedence.
timeoutSec: # [number; minimum: 1, maximum: 9007199254740991] Request timeout - A
flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
extraHttpHeaders: # [array] Extra HTTP headers - Headers to add to all events
  - name: # [string] Field Name
    value: # [string] Field Value
useRoundRobinDns: # [boolean] Round-robin DNS - Enable round-robin DNS lookup. Wh
failedRequestLoggingMode: # [string] Failed request logging mode - Data to log wh
safeHeaders: # [array of strings] Safe headers - List of headers that are safe to
systemFields: # [array of strings] System fields - Fields to automatically add to
responseRetrySettings: # [array] Settings for failed HTTP requests - Automaticall
  - httpStatus: # [number; minimum: 100, maximum: 599] HTTP status code - The HTT
    initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How l
    backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base fo
    maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) -
```

```
  timeoutRetrySettings: # [object]
    timeoutRetry: # [boolean] Retry timed-out HTTP requests

    # -------------- if timeoutRetry is true ---------------

    initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How lor
    backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base for
    maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) - Th

    # -----------------------------------------------------

  responseHonorRetryAfterHeader: # [boolean] Honor Retry-After header - Honor any F
  onBackpressure: # [string] Backpressure behavior - How to handle events when all

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] File size limit - The maximum size to store in each que
  pqMaxSize: # [string] Queue size limit - The maximum disk space that the queue ca
  pqPath: # [string] Queue file path - The location for the persistent queue files.
  pqCompress: # [string] Compression - Codec to use to compress the persisted data
  pqOnBackpressure: # [string] Queue-full behavior - How to handle events when the
  pqMode: # [string] Mode - In Error mode, PQ writes events to the filesystem if th

  # -----------------------------------------------------

  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
loki_output: # [object]
  type: # [string] Output Type
  url: # [string] Loki URL - The endpoint to send logs to
  message: # [string] Logs message field - Name of the event field that contains th
  messageFormat: # [string] Message format - Format to use when sending logs to Lok

  # -------------- if messageFormat is json ---------------

  compress: # [boolean] Compress - Compress the payload body before sending

  # -----------------------------------------------------

  labels: # [array] Logs labels - List of labels to send with logs. Labels define l
    - name: # [string] Name
```

```
    value: # [string] Value
authType: # [string] Authentication type

# -------------- if authType is token ---------------

token: # [string] Auth token - Bearer token to include in the authorization heade

# ---------------------------------------------------


# -------------- if authType is textSecret ---------------

textSecret: # [string] Auth token (text secret) - Select or create a stored text

# ---------------------------------------------------


# -------------- if authType is basic ---------------

username: # [string] Username - Username for authentication
password: # [string] Password - Password (API key in Grafana Cloud domain) for au

# ---------------------------------------------------


# -------------- if authType is credentialsSecret ---------------

credentialsSecret: # [string] Credentials secret - Select or create a secret that

# ---------------------------------------------------

concurrency: # [number; minimum: 1, maximum: 32] Request concurrency - Maximum nu
maxPayloadSizeKB: # [number; minimum: 1024, maximum: 10240] Body size limit (KB)
maxPayloadEvents: # [number] Events-per-request limit - Maximum number of events
rejectUnauthorized: # [boolean] Validate server certs - Reject certificates not a
    Enabled by default. When this setting is also present in TLS Settings (Client
    that value will take precedence.
timeoutSec: # [number; minimum: 1, maximum: 9007199254740991] Request timeout - A
flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
extraHttpHeaders: # [array] Extra HTTP headers - Headers to add to all events
  - name: # [string] Field Name
    value: # [string] Field Value
useRoundRobinDns: # [boolean] Round-robin DNS - Enable round-robin DNS lookup. Wh
```

```
  failedRequestLoggingMode: # [string] Failed request logging mode - Data to log wh
  safeHeaders: # [array of strings] Safe headers - List of headers that are safe to
  responseRetrySettings: # [array] Settings for failed HTTP requests - Automatically
    - httpStatus: # [number; minimum: 100, maximum: 599] HTTP status code - The HTT
      initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How l
      backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base fo
      maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) -
  timeoutRetrySettings: # [object]
    timeoutRetry: # [boolean] Retry timed-out HTTP requests

    # -------------- if timeoutRetry is true ---------------

    initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How lon
    backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base for
    maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) - Th

    # -------------------------------------------------------

  responseHonorRetryAfterHeader: # [boolean] Honor Retry-After header - Honor any R
  systemFields: # [array of strings] System fields - Fields to automatically add to
  enableDynamicHeaders: # [boolean] Enable dynamic headers - Add per-event HTTP hea
  onBackpressure: # [string] Backpressure behavior - How to handle events when all

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] File size limit - The maximum size to store in each que
  pqMaxSize: # [string] Queue size limit - The maximum disk space that the queue ca
  pqPath: # [string] Queue file path - The location for the persistent queue files.
  pqCompress: # [string] Compression - Codec to use to compress the persisted data
  pqOnBackpressure: # [string] Queue-full behavior - How to handle events when the
  pqMode: # [string] Mode - In Error mode, PQ writes events to the filesystem if th

  # -------------------------------------------------------

  totalMemoryLimitKB: # [number] Buffer memory limit (KB) - Maximum total size of t
  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
prometheus_output: # [object]
  type: # [string] Output Type
  url: # [string] Remote Write URL - The endpoint to send metrics to
  metricRenameExpr: # [string] Metric renaming expression - JavaScript expression t
```

```
sendMetadata: # [boolean] Send metadata - Generate and send metadata (`type` and

# -------------- if sendMetadata is true ---------------

metricsFlushPeriodSec: # [number] Metadata flush period (sec) - How frequently me

# ------------------------------------------------------

systemFields: # [array of strings] System fields - Fields to automatically add to
concurrency: # [number; minimum: 1, maximum: 32] Request concurrency - Maximum nu
maxPayloadSizeKB: # [number; minimum: 1024, maximum: 10240] Body size limit (KB)
maxPayloadEvents: # [number] Events-per-request limit - Maximum number of events
rejectUnauthorized: # [boolean] Validate server certs - Reject certificates not a
    Enabled by default. When this setting is also present in TLS Settings (Client
    that value will take precedence.
timeoutSec: # [number; minimum: 1, maximum: 9007199254740991] Request timeout - A
flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
extraHttpHeaders: # [array] Extra HTTP headers - Headers to add to all events
  - name: # [string] Field Name
    value: # [string] Field Value
useRoundRobinDns: # [boolean] Round-robin DNS - Enable round-robin DNS lookup. Wh
failedRequestLoggingMode: # [string] Failed request logging mode - Data to log wh
safeHeaders: # [array of strings] Safe headers - List of headers that are safe to
responseRetrySettings: # [array] Settings for failed HTTP requests - Automaticall
  - httpStatus: # [number; minimum: 100, maximum: 599] HTTP status code - The HTT
    initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How l
    backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base fo
    maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) -
timeoutRetrySettings: # [object]
  timeoutRetry: # [boolean] Retry timed-out HTTP requests

  # -------------- if timeoutRetry is true ---------------

  initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How lon
  backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base for
  maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) - Th

  # ------------------------------------------------------

responseHonorRetryAfterHeader: # [boolean] Honor Retry-After header - Honor any R
onBackpressure: # [string] Backpressure behavior - How to handle events when all

# -------------- if onBackpressure is queue --------------
```

```
pqMaxFileSize: # [string] File size limit - The maximum size to store in each que
pqMaxSize: # [string] Queue size limit - The maximum disk space that the queue ca
pqPath: # [string] Queue file path - The location for the persistent queue files.
pqCompress: # [string] Compression - Codec to use to compress the persisted data
pqOnBackpressure: # [string] Queue-full behavior - How to handle events when the
pqMode: # [string] Mode - In Error mode, PQ writes events to the filesystem if th

# -------------------------------------------------------

authType: # [string] Authentication type - Remote Write authentication type

# -------------- if authType is basic ---------------

username: # [string] Username
password: # [string] Password

# -------------------------------------------------------


# -------------- if authType is token ---------------

token: # [string] Token - Bearer token to include in the authorization header

# -------------------------------------------------------


# -------------- if authType is credentialsSecret ---------------

credentialsSecret: # [string] Credentials secret - Select or create a secret that

# -------------------------------------------------------


# -------------- if authType is textSecret ---------------

textSecret: # [string] Token (text secret) - Select or create a stored text secre

# -------------------------------------------------------


# -------------- if authType is oauth ---------------
```

```
  loginUrl: # [string] Login URL - URL for OAuth
  secretParamName: # [string] OAuth Secret parameter name - Secret parameter name t
  secret: # [string] OAuth secret - Secret parameter value to pass in request body
  tokenAttributeName: # [string] Token attribute name - Name of the auth token attr
  spacer: # [null]
  authHeaderExpr: # [string] Authorize expression - JavaScript expression to comput
  tokenTimeoutSecs: # [number; minimum: 1, maximum: 300000] Refresh interval (secs.
  oauthParams: # [array] OAuth parameters - Additional parameters to send in the OA
    - name: # [string] Name - OAuth parameter name
      value: # [string] Value - OAuth parameter value
  oauthHeaders: # [array] OAuth headers - Additional headers to send in the OAuth l
    - name: # [string] Name - OAuth header name
      value: # [string] Value - OAuth header value

  # -------------------------------------------------------

  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
ring_output: # [object]
  type: # [string] Output Type
  format: # [string] Data format - Format of the output data.
  partitionExpr: # [string] Partitioning expression - JS expression to define how f
  maxDataSize: # [string] Data size limit - Maximum disk space allowed to be consum
  maxDataTime: # [string] Data age limit - Maximum amount of time to retain data (e
  compress: # [string] Data compression format
  destPath: # [string] Path location - Path to use to write metrics. Defaults to $C
  onBackpressure: # [string] Backpressure behavior - How to handle events when all
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
open_telemetry_output: # [object]
  type: # [string] Output Type
  protocol: # [string] Protocol - Select a transport option for OpenTelemetry

  # -------------- if protocol is http ---------------

  httpTracesEndpointOverride: # [string] Traces endpoint override - If you want to
  httpMetricsEndpointOverride: # [string] Metrics endpoint override - If you want t
  httpLogsEndpointOverride: # [string] Logs endpoint override - If you want to send
```

```
httpCompress: # [string] Compression - Type of compression to apply to messages s
keepAlive: # [boolean] Keep alive - Disable to close the connection immediately a
rejectUnauthorized: # [boolean] Validate server certs - Reject certificates not a
    Enabled by default. When this setting is also present in TLS Settings (Client
    that value will take precedence.
useRoundRobinDns: # [boolean] Round-robin DNS - Enable round-robin DNS lookup. Wh
extraHttpHeaders: # [array] Extra HTTP headers - Headers to add to all events
  - name: # [string] Field Name
    value: # [string] Field Value
safeHeaders: # [array of strings] Safe headers - List of headers that are safe to
responseRetrySettings: # [array] Settings for failed HTTP requests - Automatically
  - httpStatus: # [number; minimum: 100, maximum: 599] HTTP status code - The HTT
    initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How l
    backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base fo
    maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) -
timeoutRetrySettings: # [object]
  timeoutRetry: # [boolean] Retry timed-out HTTP requests

  # -------------- if timeoutRetry is true ---------------

  initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How lon
  backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base for
  maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) - Th

  # -------------------------------------------------------

responseHonorRetryAfterHeader: # [boolean] Honor Retry-After header - Honor any R

# -------------------------------------------------------


# -------------- if protocol is grpc ---------------

connectionTimeout: # [number] Connection timeout - Amount of time (milliseconds)
keepAliveTime: # [number; minimum: 1] Keep alive time (seconds) - How often the s
metadata: # [array] Metadata - List of key-value pairs to send with each gRPC req
  - key: # [string] Key
    value: # [string] Value
tls: # [object] TLS settings (client side)
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------
```

```
  rejectUnauthorized: # [boolean] Validate server certs - Reject certificates tha
                trusted CA (such as the system's). Defaults to Enabled. Overrides
  certificateName: # [string] Certificate - The name of the predefined certificat
  caPath: # [string] CA certificate path - Path on client in which to find CA cer
  privKeyPath: # [string] Private key path (mutual auth) - Path on client in whic
  certPath: # [string] Certificate path (mutual auth) - Path on client in which t
  passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
  minVersion: # [string] Minimum TLS version
  maxVersion: # [string] Maximum TLS version

  # ----------------------------------------------------------

compress: # [string] Compression - Type of compression to apply to messages sent

# ----------------------------------------------------------

endpoint: # [string] Endpoint - The endpoint where OTel events will be sent. Ente
otlpVersion: # [string] OTLP version - The version of OTLP Protobuf definitions t
authType: # [string] Authentication type - OpenTelemetry authentication type

# -------------- if authType is basic ---------------

username: # [string] Username
password: # [string] Password

# ----------------------------------------------------------


# -------------- if authType is token ---------------

token: # [string] Token - Bearer token to include in the authorization header

# ----------------------------------------------------------


# -------------- if authType is credentialsSecret ---------------

credentialsSecret: # [string] Credentials secret - Select or create a secret that

# ----------------------------------------------------------


# -------------- if authType is textSecret ---------------
```

```
textSecret: # [string] Token (text secret) - Select or create a stored text secre

# -------------------------------------------------------


# -------------- if authType is oauth ---------------

loginUrl: # [string] Login URL - URL for OAuth
secretParamName: # [string] OAuth Secret parameter name - Secret parameter name t
secret: # [string] OAuth secret - Secret parameter value to pass in request body
tokenAttributeName: # [string] Token attribute name - Name of the auth token attr
spacer: # [null]
authHeaderExpr: # [string] Authorize expression - JavaScript expression to comput
tokenTimeoutSecs: # [number; minimum: 1, maximum: 300000] Refresh interval (secs.
oauthParams: # [array] OAuth parameters - Additional parameters to send in the OA
  - name: # [string] Name - OAuth parameter name
    value: # [string] Value - OAuth parameter value
oauthHeaders: # [array] OAuth headers - Additional headers to send in the OAuth t
  - name: # [string] Name - OAuth header name
    value: # [string] Value - OAuth header value

# -------------------------------------------------------

concurrency: # [number; minimum: 1, maximum: 32] Request concurrency - Maximum nu
maxPayloadSizeKB: # [number; minimum: 1024, maximum: 10240] Body size limit (KB)
timeoutSec: # [number; minimum: 1, maximum: 9007199254740991] Request timeout - A
flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
failedRequestLoggingMode: # [string] Failed request logging mode - Data to log wh
onBackpressure: # [string] Backpressure behavior - How to handle events when all

# -------------- if onBackpressure is queue ---------------

pqMaxFileSize: # [string] File size limit - The maximum size to store in each que
pqMaxSize: # [string] Queue size limit - The maximum disk space that the queue ca
pqPath: # [string] Queue file path - The location for the persistent queue files.
pqCompress: # [string] Compression - Codec to use to compress the persisted data
pqOnBackpressure: # [string] Queue-full behavior - How to handle events when the
pqMode: # [string] Mode - In Error mode, PQ writes events to the filesystem if th

# -------------------------------------------------------

description: # [string] Description
```

```
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
service_now_output: # [object]
  type: # [string] Output Type
  endpoint: # [string] Endpoint - The endpoint where ServiceNow events will be sent
  tokenSecret: # [string] [required] Auth token (text secret) - Select or create a
  authTokenName: # [string] Auth token name
  otlpVersion: # [string] [required] OTLP version - The version of OTLP Protobuf de
  maxPayloadSizeKB: # [number; minimum: 1024, maximum: 10240] Body size limit (KB)
  protocol: # [string] [required] Protocol - Select a transport option for OpenTele

  # -------------- if protocol is http ---------------

  httpTracesEndpointOverride: # [string] Traces endpoint override - If you want to
  httpMetricsEndpointOverride: # [string] Metrics endpoint override - If you want t
  httpLogsEndpointOverride: # [string] Logs endpoint override - If you want to send
  httpCompress: # [string] Compression - Type of compression to apply to messages s
  keepAlive: # [boolean] Keep alive - Disable to close the connection immediately a
  rejectUnauthorized: # [boolean] Validate server certs - Reject certificates not a
      Enabled by default. When this setting is also present in TLS Settings (Client
      that value will take precedence.
  useRoundRobinDns: # [boolean] Round-robin DNS - Enable round-robin DNS lookup. Wh
  extraHttpHeaders: # [array] Extra HTTP headers - Headers to add to all events
    - name: # [string] Field Name
      value: # [string] Field Value
  safeHeaders: # [array of strings] Safe headers - List of headers that are safe to
  responseRetrySettings: # [array] Settings for failed HTTP requests - Automatically
    - httpStatus: # [number; minimum: 100, maximum: 599] HTTP status code - The HTT
      initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How l
      backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base fo
      maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) -
  timeoutRetrySettings: # [object]
    timeoutRetry: # [boolean] Retry timed-out HTTP requests

    # -------------- if timeoutRetry is true ---------------

    initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How lon
    backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base for
    maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) - Th

    # -------------------------------------------------------
```

```
responseHonorRetryAfterHeader: # [boolean] Honor Retry-After header - Honor any R

# -------------------------------------------------------


# -------------- if protocol is grpc ---------------

connectionTimeout: # [number] Connection timeout - Amount of time (milliseconds)
keepAliveTime: # [number; minimum: 1] Keep alive time (seconds) - How often the s
metadata: # [array] Metadata - List of key-value pairs to send with each gRPC req
  - key: # [string] Key
    value: # [string] Value
tls: # [object] TLS settings (client side)
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  rejectUnauthorized: # [boolean] Validate server certs - Reject certificates tha
                trusted CA (such as the system's). Defaults to Enabled. Overrides
  certificateName: # [string] Certificate - The name of the predefined certificat
  caPath: # [string] CA certificate path - Path on client in which to find CA cer
  privKeyPath: # [string] Private key path (mutual auth) - Path on client in whic
  certPath: # [string] Certificate path (mutual auth) - Path on client in which t
  passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
  minVersion: # [string] Minimum TLS version
  maxVersion: # [string] Maximum TLS version

  # -------------------------------------------------------

compress: # [string] Compression - Type of compression to apply to messages sent

# -------------------------------------------------------

concurrency: # [number; minimum: 1, maximum: 32] Request concurrency - Maximum nu
timeoutSec: # [number; minimum: 1, maximum: 9007199254740991] Request timeout - A
flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
failedRequestLoggingMode: # [string] Failed request logging mode - Data to log wh
onBackpressure: # [string] Backpressure behavior - How to handle events when all

# -------------- if onBackpressure is queue ---------------

pqMaxFileSize: # [string] File size limit - The maximum size to store in each que
```

```
  pqMaxSize: # [string] Queue size limit - The maximum disk space that the queue ca
  pqPath: # [string] Queue file path - The location for the persistent queue files.
  pqCompress: # [string] Compression - Codec to use to compress the persisted data
  pqOnBackpressure: # [string] Queue-full behavior - How to handle events when the
  pqMode: # [string] Mode - In Error mode, PQ writes events to the filesystem if th

  # ------------------------------------------------------

  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
dataset_output: # [object]
  type: # [string] Output Type
  messageField: # [string] Message field - Name of the event field that contains th
  excludeFields: # [array of strings] Exclude fields - Fields to exclude from the e
  serverHostField: # [string] Server/host field - Name of the event field that cont
  timestampField: # [string] Timestamp field - Name of the event field that contain
  defaultSeverity: # [string] Severity - Default value for event severity. If the `
  responseRetrySettings: # [array] Settings for failed HTTP requests - Automatically
    - httpStatus: # [number; minimum: 100, maximum: 599] HTTP status code - The HTT
      initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How l
      backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base fo
      maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) -
  timeoutRetrySettings: # [object]
    timeoutRetry: # [boolean] Retry timed-out HTTP requests

    # -------------- if timeoutRetry is true ---------------

    initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How lon
    backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base for
    maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) - Th

    # ------------------------------------------------------

  responseHonorRetryAfterHeader: # [boolean] Honor Retry-After header - Honor any R
  site: # [string] DataSet site - DataSet site to which events should be sent

  # -------------- if site is custom ---------------

  customUrl: # [string]
```

```
# ------------------------------------------------------

concurrency: # [number; minimum: 1, maximum: 32] Request concurrency - Maximum nu
maxPayloadSizeKB: # [number; minimum: 1024, maximum: 6144] Body size limit (KB) -
maxPayloadEvents: # [number] Events-per-request limit - Maximum number of events
compress: # [boolean] Compress - Compress the payload body before sending
rejectUnauthorized: # [boolean] Validate server certs - Reject certificates not a
    Enabled by default. When this setting is also present in TLS Settings (Client
    that value will take precedence.
timeoutSec: # [number; minimum: 1, maximum: 9007199254740991] Request timeout - A
flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
extraHttpHeaders: # [array] Extra HTTP headers - Headers to add to all events
  - name: # [string] Field Name
    value: # [string] Field Value
useRoundRobinDns: # [boolean] Round-robin DNS - Enable round-robin DNS lookup. Wh
failedRequestLoggingMode: # [string] Failed request logging mode - Data to log wh
safeHeaders: # [array of strings] Safe headers - List of headers that are safe to
streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
onBackpressure: # [string] Backpressure behavior - How to handle events when all

# -------------- if onBackpressure is queue ---------------

pqMaxFileSize: # [string] File size limit - The maximum size to store in each que
pqMaxSize: # [string] Queue size limit - The maximum disk space that the queue ca
pqPath: # [string] Queue file path - The location for the persistent queue files.
pqCompress: # [string] Compression - Codec to use to compress the persisted data
pqOnBackpressure: # [string] Queue-full behavior - How to handle events when the
pqMode: # [string] Mode - In Error mode, PQ writes events to the filesystem if th

# -------------------------------------------------------

authType: # [string] Authentication method - Enter API key directly, or select a
apiKey: # [string] API key - A 'Log Write Access' API key for the DataSet account

# -------------- if authType is manual ---------------


# -------------------------------------------------------

textSecret: # [string] API key (text secret) - Select or create a stored text sec

# -------------- if authType is secret ---------------
```

```
  # ------------------------------------------------------

  totalMemoryLimitKB: # [number] Buffer memory limit (KB) - Maximum total size of t
  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
cribl_tcp_output: # [object]
  type: # [string] Output Type
  loadBalanced: # [boolean] Load balancing - Use load-balanced destinations

  # -------------- if loadBalanced is false ---------------

  host: # [string] Address - The hostname of the receiver
  port: # [number; maximum: 65535] Port - The port to connect to on the provided ho

  # ------------------------------------------------------


  # -------------- if loadBalanced is true ---------------

  excludeSelf: # [boolean] Exclude current host IPs - Exclude all IPs of the curren
  hosts: # [array] Destinations - Set of hosts to load-balance data to
    - host: # [string] Address - The hostname of the receiver
      port: # [number; maximum: 65535] Port - The port to connect to on the provide
      tls: # [string] TLS - Whether to inherit TLS configs from group setting or di
      servername: # [string] TLS Servername - Servername to use if establishing a T
      weight: # [number] Load Weight - Assign a weight (>0) to each endpoint to inc
  dnsResolvePeriodSec: # [number; maximum: 86400] DNS resolution period (seconds) -
  loadBalanceStatsPeriodSec: # [number; minimum: 10] Load balance stats period (sec
  maxConcurrentSenders: # [number] Connection limit - Maximum number of concurrent

  # ------------------------------------------------------

  compression: # [string] Compression - Codec to use to compress the data before se
  logFailedRequests: # [boolean] Log failed requests to disk - Use to troubleshoot
  throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to throttl
  tls: # [object] TLS settings (client side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------
```

```
    rejectUnauthorized: # [boolean] Validate server certs - Reject certificates tha
                        trusted CA (such as the system's). Defaults to Enabled. Overrides
    servername: # [string] Server name (SNI) - Server name for the SNI (Server Name
    certificateName: # [string] Certificate - The name of the predefined certificat
    caPath: # [string] CA certificate path - Path on client in which to find CA cer
    privKeyPath: # [string] Private key path (mutual auth) - Path on client in whic
    certPath: # [string] Certificate path (mutual auth) - Path on client in which t
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
    minVersion: # [string] Minimum TLS version
    maxVersion: # [string] Maximum TLS version

    # ------------------------------------------------------

  connectionTimeout: # [number] Connection timeout - Amount of time (milliseconds)
  writeTimeout: # [number] Write timeout - Amount of time (milliseconds) to wait fo
  tokenTTLMinutes: # [number; minimum: 1, maximum: 60] Auth Token TTL minutes - The
  excludeFields: # [array of strings] Exclude fields - Fields to exclude from the e
  onBackpressure: # [string] Backpressure behavior - How to handle events when all

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] File size limit - The maximum size to store in each que
  pqMaxSize: # [string] Queue size limit - The maximum disk space that the queue ca
  pqPath: # [string] Queue file path - The location for the persistent queue files.
  pqCompress: # [string] Compression - Codec to use to compress the persisted data
  pqOnBackpressure: # [string] Queue-full behavior - How to handle events when the
  pqMode: # [string] Mode - In Error mode, PQ writes events to the filesystem if th

  # -------------------------------------------------------

  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
cribl_http_output: # [object]
  type: # [string] Output Type
  loadBalanced: # [boolean] Load balancing - For optimal performance, enable load b

  # -------------- if loadBalanced is false ---------------

  url: # [string] Cribl endpoint - URL of a Cribl Worker to send events to, such as
  useRoundRobinDns: # [boolean] Round-robin DNS - Enable round-robin DNS lookup. Wh
```

```
# ------------------------------------------------------


# -------------- if loadBalanced is true ---------------

excludeSelf: # [boolean] Exclude current host IPs - Exclude all IPs of the currer
urls: # [array] Cribl Worker endpoints
  - url: # [string] Cribl Endpoint - URL of a Cribl Worker to send events to, suc
    weight: # [number] Load Weight - Assign a weight (>0) to each endpoint to inc
dnsResolvePeriodSec: # [number; maximum: 86400] DNS resolution period (seconds) -
loadBalanceStatsPeriodSec: # [number; minimum: 10] Load balance stats period (sec

# ------------------------------------------------------

tls: # [object] TLS settings (client side)
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  rejectUnauthorized: # [boolean] Validate server certs - Reject certificates tha
                trusted CA (such as the system's). Defaults to Enabled. Overrides
  servername: # [string] Server name (SNI) - Server name for the SNI (Server Name
  certificateName: # [string] Certificate - The name of the predefined certificat
  caPath: # [string] CA certificate path - Path on client in which to find CA cer
  privKeyPath: # [string] Private key path (mutual auth) - Path on client in whic
  certPath: # [string] Certificate path (mutual auth) - Path on client in which t
  passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
  minVersion: # [string] Minimum TLS version
  maxVersion: # [string] Maximum TLS version

  # ------------------------------------------------------

tokenTTLMinutes: # [number; minimum: 1, maximum: 60] Auth Token TTL minutes - The
excludeFields: # [array of strings] Exclude fields - Fields to exclude from the e
compression: # [string] Compression - Codec to use to compress the data before se
concurrency: # [number; minimum: 1, maximum: 32] Request concurrency - Maximum nu
maxPayloadSizeKB: # [number; minimum: 1024, maximum: 10240] Body size limit (KB)
maxPayloadEvents: # [number] Events-per-request limit - Maximum number of events
rejectUnauthorized: # [boolean] Validate server certs - Reject certificates not a
    Enabled by default. When this setting is also present in TLS Settings (Client
    that value will take precedence.
timeoutSec: # [number; minimum: 1, maximum: 9007199254740991] Request timeout - A
```

```
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
  extraHttpHeaders: # [array] Extra HTTP headers - Headers to add to all events
    - name: # [string] Field Name
      value: # [string] Field Value
  failedRequestLoggingMode: # [string] Failed request logging mode - Data to log wh
  safeHeaders: # [array of strings] Safe headers - List of headers that are safe to
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
  responseRetrySettings: # [array] Settings for failed HTTP requests - Automaticall
    - httpStatus: # [number; minimum: 100, maximum: 599] HTTP status code - The HTT
      initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How l
      backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base fo
      maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) -
  timeoutRetrySettings: # [object]
    timeoutRetry: # [boolean] Retry timed-out HTTP requests

    # -------------- if timeoutRetry is true ---------------

    initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How lon
    backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base for
    maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) - Th

    # -------------------------------------------------------

  responseHonorRetryAfterHeader: # [boolean] Honor Retry-After header - Honor any R
  onBackpressure: # [string] Backpressure behavior - How to handle events when all

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] File size limit - The maximum size to store in each que
  pqMaxSize: # [string] Queue size limit - The maximum disk space that the queue ca
  pqPath: # [string] Queue file path - The location for the persistent queue files.
  pqCompress: # [string] Compression - Codec to use to compress the persisted data
  pqOnBackpressure: # [string] Queue-full behavior - How to handle events when the
  pqMode: # [string] Mode - In Error mode, PQ writes events to the filesystem if th

  # -------------------------------------------------------

  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
humio_hec_output: # [object]
  type: # [string] Output Type
```

```
url: # [string] [required] LogScale endpoint - URL to a CrowdStrike Falcon LogSca
concurrency: # [number; minimum: 1, maximum: 32] Request concurrency - Maximum nu
maxPayloadSizeKB: # [number; minimum: 1024, maximum: 32768] Body size limit (KB)
maxPayloadEvents: # [number] Events-per-request limit - Maximum number of events
compress: # [boolean] Compress - Compress the payload body before sending
rejectUnauthorized: # [boolean] Validate server certs - Reject certificates not a
    Enabled by default. When this setting is also present in TLS Settings (Client
    that value will take precedence.
timeoutSec: # [number; minimum: 1, maximum: 9007199254740991] Request timeout - A
flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
extraHttpHeaders: # [array] Extra HTTP headers - Headers to add to all events
  - name: # [string] Field Name
    value: # [string] Field Value
useRoundRobinDns: # [boolean] Round-robin DNS - Enable round-robin DNS lookup. Wh
failedRequestLoggingMode: # [string] Failed request logging mode - Data to log wh
safeHeaders: # [array of strings] Safe headers - List of headers that are safe to
format: # [string] Request format - When set to JSON, the event is automatically
authType: # [string] Authentication method - Select Manual to enter an auth token

# -------------- if authType is manual ---------------

token: # [string] LogScale auth token - CrowdStrike Falcon LogScale authenticatio

# ----------------------------------------------------


# -------------- if authType is secret ---------------

textSecret: # [string] LogScale auth token (text secret) - Select or create a sto

# ----------------------------------------------------

responseRetrySettings: # [array] Settings for failed HTTP requests - Automatical
  - httpStatus: # [number; minimum: 100, maximum: 599] HTTP status code - The HTT
    initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How l
    backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base fo
    maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) -
timeoutRetrySettings: # [object]
  timeoutRetry: # [boolean] Retry timed-out HTTP requests

  # -------------- if timeoutRetry is true ---------------

  initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How lon
```

```
      backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base for
      maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) - Th

      # ------------------------------------------------------

    responseHonorRetryAfterHeader: # [boolean] Honor Retry-After header - Honor any R
    onBackpressure: # [string] Backpressure behavior - How to handle events when all

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] File size limit - The maximum size to store in each que
    pqMaxSize: # [string] Queue size limit - The maximum disk space that the queue ca
    pqPath: # [string] Queue file path - The location for the persistent queue files.
    pqCompress: # [string] Compression - Codec to use to compress the persisted data
    pqOnBackpressure: # [string] Queue-full behavior - How to handle events when the
    pqMode: # [string] Mode - In Error mode, PQ writes events to the filesystem if th

    # -------------------------------------------------------

    description: # [string] Description
    pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
    systemFields: # [array of strings] System fields - Fields to automatically add to
    environment: # [string] Environment - Optionally, enable this config only on a sp
    streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
  crowdstrike_next_gen_siem_output: # [object]
    type: # [string] Output Type
    url: # [string] [required] Next-Gen SIEM endpoint - URL provided from a CrowdStri
Example: https://ingest.<region>.crowdstrike.com/api/ingest/hec/<connection-id>/v1/se
    concurrency: # [number; minimum: 1, maximum: 32] Request concurrency - Maximum nu
    maxPayloadSizeKB: # [number; minimum: 1024, maximum: 32768] Body size limit (KB)
    maxPayloadEvents: # [number] Events-per-request limit - Maximum number of events
    compress: # [boolean] Compress - Compress the payload body before sending
    rejectUnauthorized: # [boolean] Validate server certs - Reject certificates not a
        Enabled by default. When this setting is also present in TLS Settings (Client
        that value will take precedence.
    timeoutSec: # [number; minimum: 1, maximum: 9007199254740991] Request timeout - A
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
    extraHttpHeaders: # [array] Extra HTTP headers - Headers to add to all events
      - name: # [string] Field Name
        value: # [string] Field Value
    useRoundRobinDns: # [boolean] Round-robin DNS - Enable round-robin DNS lookup. Wh
    failedRequestLoggingMode: # [string] Failed request logging mode - Data to log wh
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe to
```

```
format: # [string] Request format - When set to JSON, the event is automatically
authType: # [string] Authentication method - Select Manual to enter an auth toker

# -------------- if authType is manual ---------------

token: # [string] Next-Gen SIEM authentication token

# ----------------------------------------------------


# -------------- if authType is secret ---------------

textSecret: # [string] Next-Gen SIEM authentication token (text secret) - Select

# ----------------------------------------------------

responseRetrySettings: # [array] Settings for failed HTTP requests - Automaticall
  - httpStatus: # [number; minimum: 100, maximum: 599] HTTP status code - The HTT
    initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How l
    backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base fo
    maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) -
timeoutRetrySettings: # [object]
  timeoutRetry: # [boolean] Retry timed-out HTTP requests

  # -------------- if timeoutRetry is true ---------------

  initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How lor
  backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base for
  maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) - Th

  # ----------------------------------------------------

responseHonorRetryAfterHeader: # [boolean] Honor Retry-After header - Honor any R
onBackpressure: # [string] Backpressure behavior - How to handle events when all

# -------------- if onBackpressure is queue ---------------

pqMaxFileSize: # [string] File size limit - The maximum size to store in each que
pqMaxSize: # [string] Queue size limit - The maximum disk space that the queue ca
pqPath: # [string] Queue file path - The location for the persistent queue files.
pqCompress: # [string] Compression - Codec to use to compress the persisted data
pqOnBackpressure: # [string] Queue-full behavior - How to handle events when the
pqMode: # [string] Mode - In Error mode, PQ writes events to the filesystem if th
```

```
  # ------------------------------------------------------

  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
dl_s3_output: # [object]
  type: # [string] Output Type
  bucket: # [string] S3 bucket name - Name of the destination S3 bucket. Must be a
  region: # [string] Region - Region where the S3 bucket is located
  awsSecretKey: # [string] Secret key - Secret key. This value can be a constant or
  awsAuthenticationMethod: # [string] Authentication method - AWS authentication me

  # -------------- if awsAuthenticationMethod is manual ---------------

  awsApiKey: # [string] Access key - This value can be a constant or a JavaScript e

  # ------------------------------------------------------


  # -------------- if awsAuthenticationMethod is secret ---------------

  awsSecret: # [string] Secret key pair - Select or create a stored secret that ref

  # ------------------------------------------------------

  endpoint: # [string] Endpoint - S3 service endpoint. If empty, defaults to the AW
  signatureVersion: # [string] Signature version - Signature version to use for sig
  reuseConnections: # [boolean] Reuse connections - Reuse connections between reque
  rejectUnauthorized: # [boolean] Reject unauthorized certificates - Reject certifi
  enableAssumeRole: # [boolean] Enable for S3 - Use Assume Role credentials to acce
  assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the role
  assumeRoleExternalId: # [string] External ID - External ID to use when assuming r
  durationSeconds: # [number; minimum: 900, maximum: 43200] Duration (seconds) - Du
  stagePath: # [string] [required] Staging location - Filesystem location in which
  addIdToStagePath: # [boolean] Add output ID - Add the Output ID value to staging
  destPath: # [string] Key prefix - Prefix to prepend to files before uploading. Mu
  objectACL: # [string] Object ACL - Object ACL to assign to uploaded objects
  storageClass: # [string] Storage class - Storage class to select for uploaded obj
  serverSideEncryption: # [string] Server-side encryption for uploaded objects
```

```
# -------------- if serverSideEncryption is aws:kms ---------------

kmsKeyId: # [string] KMS key ID - ID or ARN of the KMS customer-managed key to us

# ------------------------------------------------------

removeEmptyDirs: # [boolean] Remove empty staging directories - Remove empty stag

# -------------- if removeEmptyDirs is true ---------------

emptyDirCleanupSec: # [number; minimum: 10, maximum: 86400] Staging cleanup perio

# ------------------------------------------------------

format: # [string] Data format - Format of the output data

# -------------- if format is json ---------------

compress: # [string] Compression - Data compression format to apply to HTTP conte
compressionLevel: # [string] Compression level - Compression level to apply befor

# ------------------------------------------------------


# -------------- if format is parquet ---------------

automaticSchema: # [boolean] Automatic schema - Automatically calculate the schem
parquetVersion: # [string] Parquet version - Determines which data types are supp
parquetDataPageVersion: # [string] Data page version - Serialization format of da
parquetRowGroupLength: # [number; minimum: 1, maximum: 67108864] Group row limit
parquetPageSize: # [string] Page size - Target memory size for page segments, suc
shouldLogInvalidRows: # [boolean] Log invalid rows - Log up to 3 rows that Cribl
keyValueMetadata: # [array] Metadata (optional) - The metadata of files the Desti
  - key: # [string] Key
    value: # [string] Value
enableStatistics: # [boolean] Write statistics - Statistics profile an entire fil
enableWritePageIndex: # [boolean] Write page indexes - One page index contains st
enablePageChecksum: # [boolean] Write page checksum - Parquet tools can use the c

# ------------------------------------------------------

baseFileName: # [string] File name prefix expression - JavaScript expression to c
fileNameSuffix: # [string] File name suffix expression - JavaScript expression to
```

```
  maxFileSizeMB: # [number; minimum: 5, maximum: 1024] File size limit (MB) - Maxim
  maxOpenFiles: # [number; minimum: 10, maximum: 2000] Open file limit - Maximum nu
  headerLine: # [string] Header line - If set, this line will be written to the beg
  writeHighWaterMark: # [number; minimum: 16, maximum: 4096] Writing high watermark
  onBackpressure: # [string] Backpressure behavior - How to handle events when all
  deadletterEnabled: # [boolean] Enable dead-lettering - If a file fails to move to

  # -------------- if deadletterEnabled is true ---------------

  deadletterPath: # [string] Dead-letter location - Storage location for files that
  maxRetryNum: # [number; minimum: 1] Retry limit - The maximum number of times a f

  # ------------------------------------------------------

  onDiskFullBackpressure: # [string] Disk space protection - How to handle events w
  maxFileOpenTimeSec: # [number; minimum: 10, maximum: 86400] File open time limit
  maxFileIdleTimeSec: # [number; minimum: 5, maximum: 86400] Idle time limit (sec)
  maxConcurrentFileParts: # [number; minimum: 1, maximum: 10] Concurrent file parts
  verifyPermissions: # [boolean] Verify if bucket exists - Disable if you can acces
  maxClosingFilesToBackpressure: # [number; minimum: 10, maximum: 4200] Staging fil
  partitioningFields: # [array of strings] Partition by fields - List of fields to
  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
security_lake_output: # [object]
  type: # [string] Output Type
  bucket: # [string] S3 bucket name - Name of the destination S3 bucket. Must be a
  region: # [string] [required] Region - Region where the Amazon Security Lake is l
  awsSecretKey: # [string] Secret key
  awsAuthenticationMethod: # [string] Authentication method - AWS authentication me

  # -------------- if awsAuthenticationMethod is manual ---------------

  awsApiKey: # [string] Access key - This value can be a constant or a JavaScript e

  # ------------------------------------------------------


  # -------------- if awsAuthenticationMethod is secret ---------------

  awsSecret: # [string] Secret key pair - Select or create a stored secret that ref
```

```
# ------------------------------------------------------

endpoint: # [string] Endpoint - Amazon Security Lake service endpoint. If empty,
signatureVersion: # [string] Signature version - Signature version to use for sig
reuseConnections: # [boolean] Reuse connections - Reuse connections between reque
rejectUnauthorized: # [boolean] Reject unauthorized certificates - Reject certifi
enableAssumeRole: # [boolean] Enable for S3 - Use Assume Role credentials to acce
assumeRoleArn: # [string] [required] AssumeRole ARN - Amazon Resource Name (ARN)
assumeRoleExternalId: # [string] External ID - External ID to use when assuming r
durationSeconds: # [number; minimum: 900, maximum: 43200] Duration (seconds) - Du
stagePath: # [string] [required] Staging location - Filesystem location in which
addIdToStagePath: # [boolean] Add output ID - Add the Output ID value to staging
objectACL: # [string] Object ACL - Object ACL to assign to uploaded objects
storageClass: # [string] Storage class - Storage class to select for uploaded obj
serverSideEncryption: # [string] Server-side encryption for uploaded objects

# -------------- if serverSideEncryption is aws:kms ---------------

kmsKeyId: # [string] KMS key ID - ID or ARN of the KMS customer-managed key to us

# ------------------------------------------------------

removeEmptyDirs: # [boolean] Remove empty staging directories - Remove empty stag

# -------------- if removeEmptyDirs is true ---------------

emptyDirCleanupSec: # [number; minimum: 10, maximum: 86400] Staging cleanup perio

# ------------------------------------------------------

baseFileName: # [string] File name prefix expression - JavaScript expression to c
maxFileSizeMB: # [number; minimum: 5, maximum: 1024] File size limit (MB) - Maxim
maxOpenFiles: # [number; minimum: 10, maximum: 2000] Open file limit - Maximum nu
headerLine: # [string] Header line - If set, this line will be written to the beg
writeHighWaterMark: # [number; minimum: 16, maximum: 4096] Writing high watermark
onBackpressure: # [string] Backpressure behavior - How to handle events when all
deadletterEnabled: # [boolean] Enable dead-lettering - If a file fails to move to

# -------------- if deadletterEnabled is true ---------------

deadletterPath: # [string] Dead-letter location - Storage location for files that
maxRetryNum: # [number; minimum: 1] Retry limit - The maximum number of times a f
```

```
  # ------------------------------------------------------

  onDiskFullBackpressure: # [string] Disk space protection - How to handle events w
  maxFileOpenTimeSec: # [number; minimum: 10, maximum: 86400] File open time limit
  maxFileIdleTimeSec: # [number; minimum: 5, maximum: 86400] Idle time limit (sec)
  maxConcurrentFileParts: # [number; minimum: 1, maximum: 10] Concurrent file parts
  verifyPermissions: # [boolean] Verify if bucket exists - Disable if you can acces
  maxClosingFilesToBackpressure: # [number; minimum: 10, maximum: 4200] Staging fil
  accountId: # [string] [required] Account ID - ID of the AWS account whose data th
  customSource: # [string] [required] Custom source name - Name of the custom sourc
  automaticSchema: # [boolean] Automatic schema - Automatically calculate the schem

  # -------------- if automaticSchema is false ---------------

  parquetSchema: # [string] Parquet schema - To add a new schema, navigate to Proce

  # ------------------------------------------------------

  parquetVersion: # [string] Parquet version - Determines which data types are supp
  parquetDataPageVersion: # [string] Data page version - Serialization format of da
  parquetRowGroupLength: # [number; minimum: 1, maximum: 67108864] Group row limit
  parquetPageSize: # [string] Page size - Target memory size for page segments, suc
  shouldLogInvalidRows: # [boolean] Log invalid rows - Log up to 3 rows that Cribl
  keyValueMetadata: # [array] Metadata (optional) - The metadata of files the Desti
    - key: # [string] Key
      value: # [string] Value
  enableStatistics: # [boolean] Write statistics - Statistics profile an entire fil
  enableWritePageIndex: # [boolean] Write page indexes - One page index contains st
  enablePageChecksum: # [boolean] Write page checksum - Parquet tools can use the c
  systemFields: # [array of strings] System fields - Fields to automatically add to
  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
cribl_lake_output: # [object]
  type: # [string] Output Type
  bucket: # [string] S3 bucket name - Name of the destination S3 bucket. Must be a
  region: # [string] Region - Region where the S3 bucket is located
  awsSecretKey: # [string] Secret key - Secret key. This value can be a constant or
  endpoint: # [string] Endpoint - S3 service endpoint. If empty, defaults to the AW
  signatureVersion: # [string] Signature version - Signature version to use for sig
  reuseConnections: # [boolean] Reuse connections - Reuse connections between reque
```

```
rejectUnauthorized: # [boolean] Reject unauthorized certificates - Reject certifi
enableAssumeRole: # [boolean] Enable for S3 - Use Assume Role credentials to acce
assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the role
assumeRoleExternalId: # [string] External ID - External ID to use when assuming r
durationSeconds: # [number; minimum: 900, maximum: 43200] Duration (seconds) - Du
stagePath: # [string] Staging location - Filesystem location in which to buffer f
addIdToStagePath: # [boolean] Add output ID - Add the Output ID value to staging
destPath: # [string] Lake dataset - Lake dataset to send the data to.
objectACL: # [string] Object ACL - Object ACL to assign to uploaded objects
storageClass: # [string] Storage class - Storage class to select for uploaded obj
serverSideEncryption: # [string] Server-side encryption for uploaded objects

# -------------- if serverSideEncryption is aws:kms ---------------

kmsKeyId: # [string] KMS key ID - ID or ARN of the KMS customer-managed key to us

# ------------------------------------------------------

removeEmptyDirs: # [boolean] Remove empty staging directories - Remove empty stag

# -------------- if removeEmptyDirs is true ---------------

emptyDirCleanupSec: # [number; minimum: 10, maximum: 86400] Staging cleanup perio

# ------------------------------------------------------

baseFileName: # [string] File name prefix expression - JavaScript expression to c
fileNameSuffix: # [string] File name suffix expression - JavaScript expression to
maxFileSizeMB: # [number; minimum: 5, maximum: 1024] File size limit (MB) - Maxim
maxOpenFiles: # [number; minimum: 10, maximum: 2000] Open file limit - Maximum nu
headerLine: # [string] Header line - If set, this line will be written to the beg
writeHighWaterMark: # [number; minimum: 16, maximum: 4096] Writing high watermark
onBackpressure: # [string] Backpressure behavior - How to handle events when all
deadletterEnabled: # [boolean] Enable dead-lettering - If a file fails to move to

# -------------- if deadletterEnabled is true ---------------

deadletterPath: # [string] Dead-letter location - Storage location for files that
maxRetryNum: # [number; minimum: 1] Retry limit - The maximum number of times a f

# ------------------------------------------------------

onDiskFullBackpressure: # [string] Disk space protection - How to handle events w
```

```
  maxFileOpenTimeSec: # [number; minimum: 10, maximum: 86400] File open time limit
  maxFileIdleTimeSec: # [number; minimum: 5, maximum: 86400] Idle time limit (sec)
  verifyPermissions: # [boolean] Verify if bucket exists - Disable if you can acces
  maxClosingFilesToBackpressure: # [number; minimum: 10, maximum: 4200] Staging fil
  awsAuthenticationMethod: # [string]
  format: # [string]
  maxConcurrentFileParts: # [number; minimum: 1, maximum: 10] Concurrent file parts
  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
disk_spool_output: # [object]
  type: # [string] Output Type
  timeWindow: # [string] Bucket time span - Time period for grouping spooled events
  maxDataSize: # [string] Data size limit - Maximum disk space that can be consumed
  maxDataTime: # [string] Data age limit - Maximum amount of time to retain data be
  compress: # [string] Compression - Data compression format. Default is gzip.
  partitionExpr: # [string] Partitioning expression - JavaScript expression definin
  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
click_house_output: # [object]
  type: # [string] Output Type
  url: # [string] [required] URL - URL of the ClickHouse instance. Example: http://
  authType: # [string] Authentication type

  # -------------- if authType is basic ---------------

  username: # [string] Username
  password: # [string] Password

  # ---------------------------------------------------------


  # -------------- if authType is token ---------------

  token: # [string] Token - Bearer token to include in the authorization header

  # ---------------------------------------------------------
```

```
# -------------- if authType is credentialsSecret ---------------

credentialsSecret: # [string] Credentials secret - Select or create a secret that

# ----------------------------------------------------------


# -------------- if authType is textSecret ---------------

textSecret: # [string] Token (text secret) - Select or create a stored text secre

# ----------------------------------------------------------


# -------------- if authType is oauth ---------------

loginUrl: # [string] Login URL - URL for OAuth
secretParamName: # [string] OAuth Secret parameter name - Secret parameter name t
secret: # [string] OAuth secret - Secret parameter value to pass in request body
tokenAttributeName: # [string] Token attribute name - Name of the auth token attr
spacer: # [null]
authHeaderExpr: # [string] Authorize expression - JavaScript expression to comput
tokenTimeoutSecs: # [number; minimum: 1, maximum: 300000] Refresh interval (secs.
oauthParams: # [array] OAuth parameters - Additional parameters to send in the OA
  - name: # [string] Name - OAuth parameter name
    value: # [string] Value - OAuth parameter value
oauthHeaders: # [array] OAuth headers - Additional headers to send in the OAuth t
  - name: # [string] Name - OAuth header name
    value: # [string] Value - OAuth header value

# ----------------------------------------------------------


# -------------- if authType is sslUserCertificate ---------------

sqlUsername: # [string] Username - Username for certificate authentication

# ----------------------------------------------------------

database: # [string] ClickHouse database
tableName: # [string] [required] ClickHouse table - Name of the ClickHouse table
format: # [string] Format - Data format to use when sending data to ClickHouse. D
```

```
mappingType: # [string] Mapping type - How event fields are mapped to ClickHouse

# -------------- if mappingType is automatic ---------------

excludeMappingFields: # [array of strings] Exclude fields - Fields to exclude fro

# -----------------------------------------------------------


# -------------- if mappingType is custom ---------------

describeTable: # [string] Retrieve table columns - Retrieves the table schema fro
columnMappings: # [array] Column Mapping
  - columnName: # [string] Column name - Name of the column in ClickHouse that wi
    columnType: # [string] Column type - Type of the column in the ClickHouse dat
    columnValueExpression: # [string] Column value - JavaScript expression to com

# -----------------------------------------------------------

asyncInserts: # [boolean] Async inserts - Collect data into batches for later pro

# -------------- if asyncInserts is true ---------------

waitForAsyncInserts: # [boolean] Wait for async inserts - Cribl will wait for cor

# -----------------------------------------------------------

tls: # [object] TLS settings (client side)
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  servername: # [string] Server name (SNI) - Server name for the SNI (Server Name
  certificateName: # [string] Certificate - The name of the predefined certificat
  caPath: # [string] CA certificate path - Path on client in which to find CA cer
  privKeyPath: # [string] Private key path (mutual auth) - Path on client in whic
  certPath: # [string] Certificate path (mutual auth) - Path on client in which t
  passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
  minVersion: # [string] Minimum TLS version
  maxVersion: # [string] Maximum TLS version

  # -----------------------------------------------------------
```

```
concurrency: # [number; minimum: 1, maximum: 32] Request concurrency - Maximum nu
maxPayloadSizeKB: # [number; minimum: 1024, maximum: 10240] Body size limit (KB)
maxPayloadEvents: # [number] Events-per-request limit - Maximum number of events
compress: # [boolean] Compress - Compress the payload body before sending
rejectUnauthorized: # [boolean] Validate server certs - Reject certificates not a
    Enabled by default. When this setting is also present in TLS Settings (Client
    that value will take precedence.
timeoutSec: # [number; minimum: 1, maximum: 9007199254740991] Request timeout - A
flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
extraHttpHeaders: # [array] Extra HTTP headers - Headers to add to all events
  - name: # [string] Field Name
    value: # [string] Field Value
useRoundRobinDns: # [boolean] Round-robin DNS - Enable round-robin DNS lookup. Wh
failedRequestLoggingMode: # [string] Failed request logging mode - Data to log wh
safeHeaders: # [array of strings] Safe headers - List of headers that are safe to
responseRetrySettings: # [array] Settings for failed HTTP requests - Automaticall
  - httpStatus: # [number; minimum: 100, maximum: 599] HTTP status code - The HTT
    initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How l
    backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base fo
    maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) -
timeoutRetrySettings: # [object]
  timeoutRetry: # [boolean] Retry timed-out HTTP requests

  # -------------- if timeoutRetry is true ---------------

  initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How lon
  backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base for
  maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) - Th

  # -------------------------------------------------------

responseHonorRetryAfterHeader: # [boolean] Honor Retry-After header - Honor any R
dumpFormatErrorsToDisk: # [boolean] Log last schema mismatch - Log the most recen
onBackpressure: # [string] Backpressure behavior - How to handle events when all

# -------------- if onBackpressure is queue ---------------

pqMaxFileSize: # [string] File size limit - The maximum size to store in each que
pqMaxSize: # [string] Queue size limit - The maximum disk space that the queue ca
pqPath: # [string] Queue file path - The location for the persistent queue files.
pqCompress: # [string] Compression - Codec to use to compress the persisted data
pqOnBackpressure: # [string] Queue-full behavior - How to handle events when the
pqMode: # [string] Mode - In Error mode, PQ writes events to the filesystem if th
```

```
  # ------------------------------------------------------

  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
xsiam_output: # [object]
  type: # [string] Output Type
  loadBalanced: # [boolean] Load balancing - Enable for optimal performance. Even i

  # -------------- if loadBalanced is false ---------------

  url: # [string] XSIAM endpoint - XSIAM endpoint URL to send events to, such as ht
  useRoundRobinDns: # [boolean] Round-robin DNS - Enable round-robin DNS lookup. Wh

  # ------------------------------------------------------


  # -------------- if loadBalanced is true ---------------

  excludeSelf: # [boolean] Exclude current host IPs - Exclude all IPs of the curren
  urls: # [array] XSIAM Endpoints
      weight: # [number] Load Weight - Assign a weight (>0) to each endpoint to inc
  dnsResolvePeriodSec: # [number; maximum: 86400] DNS resolution period (seconds) -
  loadBalanceStatsPeriodSec: # [number; minimum: 10] Load balance stats period (sec

  # ------------------------------------------------------

  concurrency: # [number; minimum: 1, maximum: 32] Request concurrency - Maximum nu
  maxPayloadSizeKB: # [number; minimum: 100, maximum: 10000] Body size limit (KB) -
  maxPayloadEvents: # [number] Events-per-request limit - Maximum number of events
  rejectUnauthorized: # [boolean] Validate server certs - Reject certificates not a
      Enabled by default. When this setting is also present in TLS Settings (Client
      that value will take precedence.
  timeoutSec: # [number; minimum: 1, maximum: 9007199254740991] Request timeout - A
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
  extraHttpHeaders: # [array] Extra HTTP headers - Headers to add to all events
    - name: # [string] Field Name
      value: # [string] Field Value
  failedRequestLoggingMode: # [string] Failed request logging mode - Data to log wh
  safeHeaders: # [array of strings] Safe headers - List of headers that are safe to
```

```
authType: # [string] Authentication method - Enter a token directly, or provide a

# -------------- if authType is token ---------------

token: # [string] Auth token - XSIAM authentication token

# ---------------------------------------------------


# -------------- if authType is secret ---------------

textSecret: # [string] Auth token (text secret) - Select or create a stored text

# ---------------------------------------------------

responseRetrySettings: # [array] Settings for failed HTTP requests - Automaticall
  - httpStatus: # [number; minimum: 100, maximum: 599] HTTP status code - The HTT
    initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How l
    backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base fo
    maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) -
timeoutRetrySettings: # [object]
  timeoutRetry: # [boolean] Retry timed-out HTTP requests

  # -------------- if timeoutRetry is true ---------------

  initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How lor
  backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base for
  maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) - Th

  # ---------------------------------------------------

responseHonorRetryAfterHeader: # [boolean] Honor Retry-After header - Honor any R
throttleRateReqPerSec: # [integer; maximum: 2000] Throttle request rate limit - M
onBackpressure: # [string] Backpressure behavior - How to handle events when all

# -------------- if onBackpressure is queue ---------------

pqMaxFileSize: # [string] File size limit - The maximum size to store in each que
pqMaxSize: # [string] Queue size limit - The maximum disk space that the queue ca
pqPath: # [string] Queue file path - The location for the persistent queue files.
pqCompress: # [string] Compression - Codec to use to compress the persisted data
pqOnBackpressure: # [string] Queue-full behavior - How to handle events when the
pqMode: # [string] Mode - In Error mode, PQ writes events to the filesystem if th
```

```
  # ------------------------------------------------------

  totalMemoryLimitKB: # [number] Buffer memory limit (KB) - Maximum total size of t
  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
netflow_output: # [object]
  type: # [string] Output Type
  hosts: # [array] NetFlow Destinations - One or more NetFlow destinations to forwa
    - host: # [string] Address - Destination host
      port: # [number; maximum: 65535] Port - Destination port, default is 2055
  dnsResolvePeriodSec: # [number; maximum: 86400] DNS resolution period (sec) - How
  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
dynatrace_http_output: # [object]
  type: # [string] Output Type
  method: # [string] Method - The method to use when sending events
  keepAlive: # [boolean] Keep alive - Disable to close the connection immediately a
  concurrency: # [number; minimum: 1, maximum: 32] Request concurrency - Maximum nu
  maxPayloadSizeKB: # [number; minimum: 1024, maximum: 5000] Body size limit (KB) -
  maxPayloadEvents: # [number; maximum: 50000] Events-per-request limit - Maximum r
  compress: # [boolean] Compress - Compress the payload body before sending
  rejectUnauthorized: # [boolean] Validate server certs - Reject certificates not a
      Enabled by default. When this setting is also present in TLS Settings (Client
      that value will take precedence.
  timeoutSec: # [number; minimum: 1, maximum: 9007199254740991] Request timeout - A
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
  extraHttpHeaders: # [array] Extra HTTP headers - Headers to add to all events. Yo
    - name: # [string] Field Name
      value: # [string] Field Value
  useRoundRobinDns: # [boolean] Round-robin DNS - Enable round-robin DNS lookup. Wh
  failedRequestLoggingMode: # [string] Failed request logging mode - Data to log wh
  safeHeaders: # [array of strings] Safe headers - List of headers that are safe to
  responseRetrySettings: # [array] Settings for failed HTTP requests - Automatical
    - httpStatus: # [number; minimum: 100, maximum: 599] HTTP status code - The HTT
      initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How l
      backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base fo
```

```
    maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) -
timeoutRetrySettings: # [object]
  timeoutRetry: # [boolean] Retry timed-out HTTP requests

  # -------------- if timeoutRetry is true ---------------

  initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How lor
  backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base for
  maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) - Th

  # -----------------------------------------------------

responseHonorRetryAfterHeader: # [boolean] Honor Retry-After header - Honor any F
onBackpressure: # [string] Backpressure behavior - How to handle events when all

# -------------- if onBackpressure is queue ---------------

pqMaxFileSize: # [string] File size limit - The maximum size to store in each que
pqMaxSize: # [string] Queue size limit - The maximum disk space that the queue ca
pqPath: # [string] Queue file path - The location for the persistent queue files.
pqCompress: # [string] Compression - Codec to use to compress the persisted data
pqOnBackpressure: # [string] Queue-full behavior - How to handle events when the
pqMode: # [string] Mode - In Error mode, PQ writes events to the filesystem if th

# -----------------------------------------------------

authType: # [string] Authentication type

# -------------- if authType is token ---------------

token: # [string] Token - Bearer token to include in the authorization header

# -----------------------------------------------------


# -------------- if authType is textSecret ---------------

textSecret: # [string] Token (text secret) - Select or create a stored text secre

# -----------------------------------------------------

format: # [string] Format - How to format events before sending. Defaults to JSON
endpoint: # [string] [required] Endpoint
```

```
  # -------------- if endpoint is cloud ---------------

  environmentId: # [string] Environment ID - ID of the environment to send to

  # ----------------------------------------------------


  # -------------- if endpoint is activeGate ---------------

  activeGateDomain: # [string] ActiveGate domain - ActiveGate domain with Log analy
  environmentId: # [string] Environment ID - ID of the environment to send to

  # ----------------------------------------------------


  # -------------- if endpoint is manual ---------------

  url: # [string] URL - URL to send events to. Can be overwritten by an event's __u

  # ----------------------------------------------------

  telemetryType: # [string] [required] Telemetry type
  totalMemoryLimitKB: # [number] Buffer memory limit (KB) - Maximum total size of t
  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
dynatrace_otlp_output: # [object]
  type: # [string] Output Type
  protocol: # [string] [required] Protocol - Select a transport option for Dynatrac

  # -------------- if protocol is http ---------------

  httpTracesEndpointOverride: # [string] Traces endpoint override - If you want to
  httpMetricsEndpointOverride: # [string] Metrics endpoint override - If you want t
  httpLogsEndpointOverride: # [string] Logs endpoint override - If you want to send
  httpCompress: # [string] Compression - Type of compression to apply to messages s
  keepAlive: # [boolean] Keep alive - Disable to close the connection immediately a
  rejectUnauthorized: # [boolean] Validate server certs - Reject certificates not a
      Enabled by default. When this setting is also present in TLS Settings (Client
      that value will take precedence.
```

```
useRoundRobinDns: # [boolean] Round-robin DNS - Enable round-robin DNS lookup. Wh
extraHttpHeaders: # [array] Extra HTTP headers - Headers to add to all events
  - name: # [string] Field Name
    value: # [string] Field Value
safeHeaders: # [array of strings] Safe headers - List of headers that are safe to
responseRetrySettings: # [array] Settings for failed HTTP requests - Automaticall
  - httpStatus: # [number; minimum: 100, maximum: 599] HTTP status code - The HTT
    initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How l
    backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base fo
    maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) -
timeoutRetrySettings: # [object]
  timeoutRetry: # [boolean] Retry timed-out HTTP requests

  # -------------- if timeoutRetry is true ---------------

  initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How lon
  backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base for
  maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) - Th

  # ------------------------------------------------------

responseHonorRetryAfterHeader: # [boolean] Honor Retry-After header - Honor any R

# -------------------------------------------------------

endpoint: # [string] Endpoint - The endpoint where Dynatrace events will be sent.
otlpVersion: # [string] [required] OTLP version - The version of OTLP Protobuf de
concurrency: # [number; minimum: 1, maximum: 32] Request concurrency - Maximum nu
maxPayloadSizeKB: # [number; minimum: 1024, maximum: 4096] Body size limit - Maxi
timeoutSec: # [number; minimum: 1, maximum: 9007199254740991] Request timeout - A
flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
failedRequestLoggingMode: # [string] Failed request logging mode - Data to log wh
endpointType: # [string] [required] Endpoint type - Select the type of Dynatrace
tokenSecret: # [string] [required] Auth token (text secret) - Select or create a
authTokenName: # [string] Api-Token name
onBackpressure: # [string] Backpressure behavior - How to handle events when all

# -------------- if onBackpressure is queue ---------------

pqMaxFileSize: # [string] File size limit - The maximum size to store in each que
pqMaxSize: # [string] Queue size limit - The maximum disk space that the queue ca
pqPath: # [string] Queue file path - The location for the persistent queue files.
pqCompress: # [string] Compression - Codec to use to compress the persisted data
```

```
  pqOnBackpressure: # [string] Queue-full behavior - How to handle events when the
  pqMode: # [string] Mode - In Error mode, PQ writes events to the filesystem if th

  # ------------------------------------------------------

  description: # [string] Description
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Fields to automatically add to
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Tags for filtering and grouping in Cribl
sentinel_one_ai_siem_output: # [object]
  type: # [string] Output Type
  region: # [string] Region - The SentinelOne region to send events to. In most cas

  # -------------- if region is Custom ---------------

  baseUrl: # [string] Base AI SIEM endpoint URL - Base URL of the endpoint used to

  # ------------------------------------------------------

  endpoint: # [string] [required] AI SIEM endpoint path - Endpoint to send events t

  # -------------- if endpoint is /services/collector/event ---------------

  hostExpression: # [string] serverHost expression - Define serverHost for events u
  sourceExpression: # [string] logFile expression - Define logFile for events using
  sourceTypeExpression: # [string] parser expression - Define the parser for events
  dataSourceCategoryExpression: # [string] dataSource.category expression - Define
  dataSourceNameExpression: # [string] dataSource.name expression - Define the data
  dataSourceVendorExpression: # [string] dataSource.vendor expression - Define the
  eventTypeExpression: # [string] event.type expression - Optionally, define the ev

  # ------------------------------------------------------



  # -------------- if endpoint is /services/collector/raw ---------------

  host: # [string] serverHost expression - Define the serverHost for events using a
  source: # [string] logFile - Specify the logFile value to pass as a parameter to
  sourceType: # [string] parser - Specify the sourcetype parameter for SentinelOne
  dataSourceCategory: # [string] dataSource.category - Specify the dataSource.categ
  dataSourceName: # [string] dataSource.name - Specify the dataSource.name value to
  dataSourceVendor: # [string] dataSource.vendor - Specify the dataSource.vendorval
```

```
eventType: # [string] event.type - Specify the event.type value to pass as an opt

# ------------------------------------------------------

concurrency: # [number; minimum: 1, maximum: 32] Request concurrency - Maximum nu
maxPayloadSizeKB: # [number; minimum: 1024, maximum: 2097152] Body size limit (KB
maxPayloadEvents: # [number] Events-per-request limit - Maximum number of events
compress: # [boolean] Compress - Compress the payload body before sending
rejectUnauthorized: # [boolean] Validate server certs - Reject certificates not a
    Enabled by default. When this setting is also present in TLS Settings (Client
    that value will take precedence.
timeoutSec: # [number; minimum: 1, maximum: 9007199254740991] Request timeout - A
flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
extraHttpHeaders: # [array] Extra HTTP headers - Headers to add to all events
  - name: # [string] Field Name
    value: # [string] Field Value
failedRequestLoggingMode: # [string] Failed request logging mode - Data to log wh
safeHeaders: # [array of strings] Safe headers - List of headers that are safe to
authType: # [string] Authentication method - Select Manual to enter an auth token

# -------------- if authType is manual ---------------

token: # [string] AI SIEM API Key - In the SentinelOne Console select Policy & Se

# ------------------------------------------------------


# -------------- if authType is secret ---------------

textSecret: # [string] AI SIEM API Key (text secret) - Select or create a stored

# ------------------------------------------------------

responseRetrySettings: # [array] Settings for failed HTTP requests - Automaticall
  - httpStatus: # [number; minimum: 100, maximum: 599] HTTP status code - The HTT
    initialBackoff: # [number; maximum: 600000] Pre-backoff interval (ms) - How l
    backoffRate: # [number; minimum: 1, maximum: 20] Backoff multiplier - Base fo
    maxBackoff: # [number; minimum: 10000, maximum: 180000] Backoff limit (ms) -
timeoutRetrySettings: # [object]
```

17.5.17. PARSERS.YML

`parsers.yml` stores configuration data for the Knowledge > Parsers Library.

parsers.yml

```
# Parser ID - Unique identifier for this parser
# [string; required]
id:
# Library - Library classification for the parser
# One of: cribl | cribl-custom | custom
# [string]
lib:
# Description - Description of the parser
# [string]
description:
# Tags - Tags associated with the parser
# [string]
tags:
# Type - Parser or formatter type to use
# One of: csv | elff | clf | kvp | json | delim | regex | grok
# [string; required]
type:
# Fields - Field names to extract
fields:
# Mode - Operation mode - Extract creates new fields. Reserialize extracts and filter
# reserializes.
# One of: extract | reserialize
# [string; default: extract]
mode:
# Source field - Field containing text to be parsed
# [string; default: _raw]
srcField:
# Destination field - Destination field name
# [string]
dstField:
# Clean fields - Clean field names by replacing non [a-zA-Z0-9] characters with _
# [boolean]
cleanFields:
# Field filter expression
# [string]
fieldFilterExpr:
# Keep - Field names to keep
keep:
# Remove - Field names to remove
remove:
# Regular expression - Regular expression for parsing
# [string]
```

```
regex:
# Regular expression list - List of regular expressions for parsing
regexList:
# Field name expression
# [string]
fieldNameExpression:
# Pattern - Grok pattern for parsing
# [string]
pattern:
# Pattern list - List of grok patterns
```

17.5.18. Persistent-Queue.Yml

`persistent-queue.yml` contains settings for configuring persistent queues.

persistent-queue.yml

```
# Buffer size limit - The maximum number of events to hold in memory before writing t
# [number; default: 1000]
maxBufferSize:
# File size limit - The maximum size to store in each queue file before closing and c
# Enter a numeral with units of KB, MB, etc.
# [string; default: 10MB]
maxFileSize:
# Queue size limit - The maximum disk space that the queue can consume before queueir
# [string; default: 1GB]
maxSize:
# Mode - Queue operation mode
# One of: always | never | smart
# [string; default: always]
mode:
# Queue file path - The location for the persistent queue files
# [string]
path:
# Compression - Codec to use to compress the persisted data
# One of: gzip | none
# [string]
compress:
# Commit frequency - The number of events to send downstream before committing that S
# [number]
commitFrequency:
```

Copy

17.5.19. Regexes.Yml

`regexes.yml` maintains a list of regexes. Cribl's Regex Library ships under `default`. Each regex is listed according to the following pattern:

regexes.yml

```
# Regex ID - Unique identifier for the regex pattern
# [string; required]
id:
# Library - Library classification for the regex
# One of: cribl | cribl-custom | custom
# [string]
lib:
# Description - Description of the regex pattern
# [string]
description:
# Regex pattern - The regular expression pattern
# [string; required]
regex:
# Sample data - Optionally, paste in sample data to match against this regex
# [string]
sampleData:
# Tags - Tags associated with the regex
# [string]
tags:
```

Copy

17.5.20. SAMPLES.YML

`samples.yml` contains metadata about about stored sample data files (size, number of events, date created, name, and so on).

samples.yml

```
# ID - Unique identifier for the sample
# [string; required]
id:
# Sample Name - Name of the sample file
# [string; required]
sampleName:
# Pipeline ID - Associate with Pipeline - Select a pipeline to associate with sample
# sure. Deprecated.
# [string]
pipelineId:
# Pack ID - ID of the pack this sample belongs to
# [string]
packId:
# Is Pack Only - Whether this sample only exists within this pack and cannot be turne
# [boolean]
isPackOnly:
# Description - Brief description of this sample file. Optional.
# [string]
description:
# Tags - One or more tags related to this sample file. Optional.
# [string]
tags:
# Library - Library classification for the sample
# [string]
lib:
# Created - Timestamp when the sample was created
# [number]
created:
# Modified - Timestamp when the sample was last modified
# [number]
```

17.5.21. Schemas.Yml

`schemas.yml` stores configuration data for the Knowledge > Schema Library.

schemas.yml

```
# Schema ID - Unique identifier for the schema
# [string; required]
id:
# Description - Description of the schema
# [string]
description:
# Schema - JSON schema matching standards of draft version 2019-09
# [string; required]
schema:
# Tags - Tags associated with the schema
# [string]
tags:
```

Copy

17.5.22. Scripts.Yml

`scripts.yml` stores configuration data for scripts configured at **Settings** > **Global** > **Scripts**:

scripts.yml

```
# Script ID - Unique identifier for the script (cannot contain forward slashes)
# [string; required]
id:
# Command - Command to execute for this script
# [string; required]
command:
# Description - Description of the script
# [string]
description:
# Arguments - Arguments to pass when executing this script
args:
# Env variables - Extra environment variables to set when executing script
env:
```

Copy

17.5.23. Service.Yml

`service.yml` maintains configuration for Cribl Edge service processes.

In the UI, you can configure some of these settings on the Leader at **Settings** > **Global** > **System** > **Service Processes** > **Services**.



Be careful about reducing the predefined `memoryLimit` values. Extremely low values can prevent some Cribl Edge components from functioning.

service.yml

```
# Service configuration - Configuration for various Cribl services
# Available service types: connections, metrics, notifications, stream_connections, e
connections:
  # Connections listener number of processes - Negative values are offsets from the r
  # on the host.
  # [number; default: 1]
  procs:
  # Connections listener heap memory limit - Heap memory limit for the connection lis
  # [string; default: 2GB]
  memoryLimit:
metrics:
  # Metrics number of processes - Single process; values above 1 will be treated as 1
  # offsets from the number of available cores on the host.
  # [number; default: 1]
  procs:
  # Metrics heap memory limit - Heap memory limit for the metrics process
  # [string; default: 2GB]
  memoryLimit:
notifications:
  # Notifications number of processes - Single process; values above 1 will be treate
  # are offsets from the number of available cores on the host.
  # [number; default: 1]
  procs:
  # Notifications heap memory limit - Heap memory limit for the notifications process
  # [string; default: 2GB]
  memoryLimit:
stream_connections:
  # Stream connection processes - Negative values are offsets from the number of avai
  # [number; default: 1]
  procs:
  # Auto scale - Do not modify this setting. Defines whether Leader can start the ser
  # Worker node connects to it.
  # [boolean]
  autoScale:
  # Stream connection memory limit - Heap memory limit for the Stream connection proc
  # [string; default: 2GB]
  memoryLimit:
edge_connections:
  # Edge connection processes - Negative values are offsets from the number of availa
  # [number; default: 1]
  procs:
  # Edge auto scale - Do not modify this setting. Defines whether Leader can start th
```

```
  # qualifying Edge node connects to it.
  # [boolean]
  autoScale:
  # Edge connection memory limit - Heap memory limit for the Edge connection processe
  # [string; default: 2GB]
  memoryLimit:
# Connection proxy service - Proxy service that runs on a standby Leader node to forw
# from port 4200 to the primary Leader.
connection_proxy:
  # Connection proxy processes - Number of processes to spawn for connection proxy se
  # Set to 0 to disable the service. Negative values are offsets from the number of a
  # [number; default: 0]
```

17.5.24. VARS.YML

`vars.yml` stores configuration data for the Knowledge > Global Variables Library.



As of Cribl Edge 4.16.0, new Fleets do not have sample variables by default.

vars.yml

```
# Name - Global variable name
# [string; required]
id:
# Library - Library classification for the variable
# One of: cribl | cribl-custom | custom
# [string]
lib:
# Description - Description of the global variable
# [string]
description:
# Type - Data type of the variable
# One of: string | number | encryptedString | boolean | array | object | expression |
# [string; default: any; required]
type:
# Value - Value or expression for the variable
# [string; required]
value:
# Tags - Tags associated with the variable
# [string]
tags:
# Arguments - For expression type variables, define function arguments
args:
```

Copy

17.6. Kubernetes Metrics Details

Consult the reference for the metrics reported by the System Metrics Source in Kubernetes.

This document provides an overview of the Kubernetes metrics collected by Cribl Edge's Kubernetes Metrics Source. These metrics are designed to align with the schema defined by the kube-state-metrics project.

Cribl Edge translates the raw metrics into a structured format while maintaining the same metric names, units, and dimensions as defined in the `kube-state-metrics` schema. This alignment helps to integrate with existing monitoring tools and workflows that rely on the standard.

The metrics are further organized into logical sections based on the Kubernetes resource types they represent:

- Auth
- Cluster
- Policy
- Service
- Storage
- Workload

Auth

Metrics related to user authentication and access control within the cluster. The metrics are further broken down into the following Metric sub-types.

`CertificateSigningRequest Metrics`

Metric name	Metric type	Description
`kube_certificatesigningrequest_annotations`	Gauge	Kubernetes annotations converted to Prometheus labels controlled via [`--metric-annotations-allowlist`] (https://github.com/kubernetes/kube-state-metrics/blob/main/docs/developer/cli-arguments.md)

Metric name	Metric type	Description
kube_certificatesigningrequest_created	Gauge	Creation timestamp (in Unix time) of the CertificateSigningRequest.
kube_certificatesigningrequest_condition	Gauge	The number of each CertificateSigningRequest condition.
kube_certificatesigningrequest_labels	Gauge	Kubernetes labels converted to Prometheus labels controlled via --metric-annotations-allowlist
kube_certificatesigningrequest_cert_length	Gauge	Length of the issued certificate.

Role Metrics

Metric Name	Metric Type	Description
kube_role_annotations	Gauge	Kubernetes annotations converted to Prometheus labels controlled via [--metric-annotations-allowlist (https://github.com/kubernetes/kube-state-metrics/blob/main/docs/developer/cli-arguments.md).
kube_role_labels	Gauge	Kubernetes labels converted to Prometheus labels controlled via --metric-labels-allowlist
kube_role_info	Gauge	Information about a Kubernetes role.
kube_role_created	Gauge	Timestamp (in Unix time) when the role was created.
kube_role_metadata_resource_version	Gauge	Resource version of the role object.

RoleBinding Metrics

Metric name	Metric type	Description
kube_rolebinding_annotations	Gauge	Kubernetes annotations converted to Prometheus labels controlled via ---

Metric name	Metric type	Description
		`metric-annotations-allowlist`
`kube_rolebinding_labels`	Gauge	Kubernetes labels converted to Prometheus labels controlled via `--metric-annotations-allowlist`
`kube_rolebinding_info`	Gauge	Information about a Kubernetes rolebinding.
`kube_rolebinding_created`	Gauge	Creation timestamp (in Unix time) of the RoleBinding.
`kube_rolebinding_metadata_resource_version`	Gauge	Resource version of the RoleBinding object.

`ServiceAccount` Metrics

Metric name	Metric type	Description
`kube_serviceaccount_info`	Gauge	Information about a service account.
`kube_serviceaccount_created`	Gauge	Creation timestamp (in Unix time) of the service account.
`kube_serviceaccount_deleted`	Gauge	Creation timestamp (in Unix time) of the service account (if applicable).
`kube_serviceaccount_secret`	Gauge	Secret being referenced by a service account.
`kube_serviceaccount_image_pull_secret`	Gauge	Secret being referenced by a service account for the purpose of pulling images.
`kube_serviceaccount_annotations`	Gauge	Kubernetes annotations converted to Prometheus labels controlled via `--metric-annotations-allowlist`
`kube_serviceaccount_labels`	Gauge	Kubernetes labels converted to Prometheus labels controlled via `--metric-annotations-allowlist`

Cluster

Metrics related to the overall health and status of the Kubernetes cluster itself, including nodes and namespaces.

`ClusterRole` Metrics

Metric name	Metric type	Description
`kube_clusterrole_annotations`	Gauge	Kubernetes annotations converted to Prometheus labels controlled via `--metric-annotations-allowlist` .
`kube_clusterrole_labels`	Gauge	Kubernetes labels converted to Prometheus labels controlled via `--metric-annotations-allowlist` .
`kube_clusterrole_info`	Gauge	Information about a Kubernetes cluster role.
`kube_clusterrole_created`	Gauge	Creation timestamp (in Unix time) of the ClusterRole.
`kube_clusterrole_metadata_resource_version`	Gauge	Resource version of the ClusterRole object.

`ClusterRoleBinding` Metrics

Metric name	Metric type	Description
`kube_clusterrolebinding_annotations`	Gauge	Kubernetes annotations converted to Prometheus labels controlled via `--metric-annotations-allowlist` .
`kube_clusterrolebinding_labels`	Gauge	Kubernetes labels converted to Prometheus labels controlled via `--metric-annotations-allowlist` .

Metric name	Metric type	Description
`kube_clusterrolebinding_info`	Gauge	Information about a Kubernetes cluster role binding.
`kube_clusterrolebinding_created`	Gauge	Creation timestamp (in Unix time) of the ClusterRoleBinding.
`kube_clusterrolebinding_metadata_resource_version`	Gauge	Resource version of the ClusterRoleBinding object.

Lease Metrics

Metric name	Metric type	Description
`kube_lease_owner`	Gauge	Information about the lease owner.
`kube_lease_renew_time`	Gauge	Time when the lease will be renewed.

Namespace Metrics

Metric name	Metric type	Description
kube_namespace_annotations	Gauge	Kubernetes annotations converted to Prometheus labels controlled via `--metric-annotations-allowlist` .
kube_namespace_created	Gauge	Creation timestamp (in Unix time) of the namespace.
kube_namespace_labels	Gauge	Kubernetes labels converted to Prometheus labels controlled via `--metric-annotations-allowlist` .
kube_namespace_status_phase	Gauge	The phase of the namespace (Active or Terminating).

Node Metrics

Metric name	Metric type	Description
kube_node_created	Gauge	Creation timestamp (in Unix time) of the node.
kube_node_status_allocatable	Gauge	The amount of resources allocatable for pods (after reserving some for system daemons).
kube_node_status_capacity	Gauge	The total amount of resources available for a node.
kube_node_status_condition	Gauge	The condition of a cluster node.
kube_node_spec_schedulable	Gauge	Whether a node can schedule new pods.
kube_node_spec_taint	Gauge	The taint of a cluster node.
kube_node_status_addresses	Gauge	The addresses of a node.
kube_node_labels	Gauge	Kubernetes labels converted to Prometheus labels controlled via `--metric-annotations-allowlist` .
kube_node_status_phase	Gauge	The phase of the node (Unknown, Running, Terminating).
kube_node_info	Gauge	Information about a cluster node.

Policy

Metrics related to resource quotas, limits, and network policies that govern resource usage within the cluster.

`LimitRange` Metrics

Metric name	Metric type	Description
kube_limitrange	Gauge	Information about resource limits and requests.
kube_limitrange_created	Gauge	Creation timestamp (in Unix time) of the LimitRange.

`Network Policy` Metrics

Metric name	Metric type	Description
kube_networkpolicy_annotations	Gauge	Kubernetes annotations converted to Prometheus labels controlled via metric-annotations-allowlist
kube_networkpolicy_created	Gauge	Creation timestamp (in Unix time) of the NetworkPolicy.
kube_networkpolicy_labels	Gauge	Kubernetes labels converted to Prometheus labels controlled via metric-labels-allowlist)
kube_networkpolicy_spec_egress_rules	Gauge	The number of egress rules defined in the NetworkPolicy.
kube_networkpolicy_spec_ingress_rules	Gauge	The number of ingress rules defined in the NetworkPolicy.

PodDisruptionBudget Metrics

Metric name	Metric type	Description
kube_poddisruptionbudget_annotations	Gauge	Kubernetes annotations converted to Prometheus labels controlled via --metric-annotations-allowlist
kube_poddisruptionbudget_labels	Gauge	Kubernetes labels converted to Prometheus labels controlled via --metric-labels-allowlist]
kube_poddisruptionbudget_created	Gauge	Creation timestamp (in Unix time) of the PodDisruptionBudget.
kube_poddisruptionbudget_status_current_healthy	Gauge	The number of currently healthy pods relative to the

Metric name	Metric type	Description
		PodDisruptionBudget's desired healthy pods.
`kube_poddisruptionbudget_status_desired_healthy`	Gauge	The minimum number of healthy pods that the PodDisruptionBudget requires.
`kube_poddisruptionbudget_status_pod_disruptions_allowed`	Gauge	The number of pod disruptions that are currently allowed for the PodDisruptionBudget.
`kube_poddisruptionbudget_status_expected_pods`	Gauge	The total number of pods that the PodDisruptionBudget expects to be running at any given time.
`kube_poddisruptionbudget_status_observed_generation`	Gauge	The generation observed by the PodDisruptionBudget controller.

`ResourceQuota` Metrics

Metric name	Metric type	Description
`kube_resourcequota`	Gauge	Provides information about the resource quotas defined for namespaces.
`kube_resourcequota_created`	Gauge	Creation timestamp (in Unix time) of the ResourceQuota.
`kube_resourcequota_annotations`	Gauge	Kubernetes annotations converted to Prometheus labels controlled via `--metric-annotations-allowlist`
`kube_resourcequota_labels`	Gauge	Kubernetes labels converted to Prometheus labels controlled via `--metric-labels-allowlist`

Service

These metrics provide information about the health and connectivity of services within the cluster.

`Endpoint` Metrics

Metric name	Metric type	Description
`kube_endpoint_annotations`	Gauge	Kubernetes annotations converted to Prometheus labels controlled via `--metric-annotations-allowlist`
`kube_endpoint_info`	Gauge	Information about a Kubernetes endpoint.
`kube_endpoint_labels`	Gauge	Kubernetes labels converted to Prometheus labels controlled via `[--metric-labels-allowlist]` ((https://github.com/kubernetes/kube-state-metrics/blob/main/docs/developer/cli-arguments.md)
`kube_endpoint_created`	Gauge	Time a Kubernetes endpoint was created.
`kube_endpoint_ports`	Gauge	Information about ports of a Kubernetes endpoint.
`kube_endpoint_address`	Gauge	IP addresses and ports of a Kubernetes endpoint.

`Endpointslice` Metrics

Metric name	Metric type	Description
kube_endpointslice_annotations	Gauge	Kubernetes annotations converted to Prometheus labels controlled via `--metric-annotations-allowlist` .
`kube_endpointslice_info`	Gauge	Information about a Kubernetes endpointslice.
`kube_endpointslice_ports`	Gauge	Information about ports of a Kubernetes endpointslice.
`kube_endpointslice_endpoints`	Gauge	Information about endpoints in a Kubernetes endpointslice.

Metric name	Metric type	Description
kube_endpointslice_endpoints_hints	Gauge	Hints applied to an endpointslice.
kube_endpointslice_labels	Gauge	Kubernetes labels converted to Prometheus labels controlled via --metric-annotations-allowlist .
kube_endpointslice_created	Gauge	Time a Kubernetes endpointslice was created.

Ingress Metrics

Metric name	Metric type	Description
kube_ingress_annotations	Gauge	Kubernetes annotations converted to Prometheus labels controlled via --metric-annotations-allowlist
kube_ingress_info	Gauge	Information about a Kubernetes ingress.
kube_ingress_labels	Gauge	Kubernetes labels converted to Prometheus labels controlled via --metric-labels-allowlist
kube_ingress_created	Gauge	Time a Kubernetes ingress was created.
kube_ingress_metadata_resource_version	Gauge	Resource version of the ingress object.
kube_ingress_path	Gauge	Information about a path served by an ingress.
kube_ingress_tls	Gauge	Information about TLS configuration of an ingress.

IngressClass Metrics

Metric name	Metric type	Description
kube_ingressclass_annotations	Gauge	Kubernetes annotations converted to Prometheus labels controlled via --metric-labels-allowlist

Metric name	Metric type	Description
kube_ingressclass_info	Gauge	Information about a Kubernetes ingressclass.
kube_ingressclass_labels	Gauge	Kubernetes labels converted to Prometheus labels controlled via --metric-labels-allowlist
kube_ingressclass_created	Gauge	Time a Kubernetes ingressclass was created.

Service Metrics

Metric name	Metric type	Description
kube_service_annotations	Gauge	Kubernetes annotations converted to Prometheus labels controlled via --metric-labels-allowlist .
kube_service_info	Gauge	Information about a service.
kube_service_labels	Gauge	Kubernetes labels converted to Prometheus labels controlled via --metric-labels-allowlist .
kube_service_created	Gauge	Creation timestamp (in Unix time) of the service.
kube_service_spec_type	Gauge	Type of the service.
kube_service_spec_external_ip	Gauge	Service external IPs. One series for each IP.
kube_service_status_load_balancer_ingress	Gauge	Service load balancer ingress status.

Storage

Metrics related to the usage and performance of storage resources within your cluster.

ConfigMap Metrics

Metric name	Metric type	Description
kube_configmap_annotations	Gauge	Kubernetes annotations converted to Prometheus labels controlled via --metric-labels-allowlist .
kube_configmap_labels	Gauge	Kubernetes labels converted to Prometheus labels controlled via --metric-labels-allowlist .
kube_configmap_info	Gauge	Information about a ConfigMap.
kube_configmap_created	Gauge	Time a ConfigMap was created.
kube_configmap_metadata_resource_version	Gauge	Resource version of the ConfigMap object.

PersistentVolume Metrics

Metric Name	Metric Type	Description
kube_persistentvolume_annotations	Gauge	Kubernetes annotations converted to Prometheus labels controlled via –metric-annotations-allowlist
kube_persistentvolume_capacity_bytes	Gauge	
kube_persistentvolume_status_phase	Gauge	
kube_persistentvolume_claim_ref	Gauge	
kube_persistentvolume_labels	Gauge	Kubernetes labels converted to Prometheus labels controlled via –metric-labels-allowlist
kube_persistentvolume_info	Gauge	Information about Persistent Volumes
kube_persistentvolume_created	Gauge	Creation timestamp (in Unix time)
kube_persistentvolume_deletion_timestamp	Gauge	Deletion timestamp (in Unix time)
kube_persistentvolume_csi_attributes	Gauge	CSI attributes of the Persistent Volume, disabled by default, manage with –metric-opt-in-list

Metric Name	Metric Type	Description
kube_persistentvolume_volume_mode	Gauge	Volume Mode information for the PersistentVolume.

PersistentVolumeClaim Metrics

Metric Name	Metric Type	Description
kube_persistentvolumeclaim_annotations	Gauge	Kubernetes annotatic converted to Prometheus labels controlled via –metric annotations-allowlist
kube_persistentvolumeclaim_access_mode	Gauge	Access modes allowe for this claim.
kube_persistentvolumeclaim_info	Gauge	Information about PersistentVolumeClai
kube_persistentvolumeclaim_labels	Gauge	Kubernetes labels converted to Prometheus labels controlled via –metric labels-allowlist
kube_persistentvolumeclaim_resource_requests_storage_bytes	Gauge	Requested storage si in bytes
kube_persistentvolumeclaim_status_condition	Gauge	Condition of the clain
kube_persistentvolumeclaim_status_phase	Gauge	Phase of the claim
kube_persistentvolumeclaim_created	Gauge	Creation timestamp (Unix time)
kube_persistentvolumeclaim_deletion_timestamp	Gauge	Deletion timestamp (Unix time)

Secret Metrics

Metric name	Metric type	Description
kube_secret_annotations	Gauge	Kubernetes annotations converted to Prometheus labels controlled via --metric-annotations-allowlist
kube_secret_info	Gauge	Information about Secrets (such as, type).
kube_secret_type	Gauge	Type of the Secret.
kube_secret_created	Gauge	Creation timestamp (in Unix time) of the Secret.
kube_secret_metadata_resource_version	Gauge	Resource version of the Secret object.
kube_secret_owner	Gauge	Owner information of the Secret.

StorageClass Metrics

Metric name	Metric type	Description
kube_storageclass_annotations	Gauge	Kubernetes annotations converted to Prometheus labels controlled via --metric-annotations-allowlist .
kube_storageclass_info	Gauge	Information about StorageClasses (such as, provisioner, reclaim policy, volume binding mode).
kube_storageclass_labels	Gauge	Kubernetes labels converted to Prometheus labels controlled via --metric-annotations-allowlist
kube_storageclass_created	Gauge	Creation timestamp (in Unix time) of the StorageClass.

VolumeAttachment Metrics

Metric name	Metric type	Description
kube_volumeattachment_info	Gauge	Information about VolumeAttachments (such as, attacher, node).

Metric name	Metric type	Description
`kube_volumeattachment_created`	Gauge	Creation timestamp (in Unix time) of the VolumeAttachment.
`kube_volumeattachment_spec_source_persistentvolume`	Gauge	PersistentVolume name used in the VolumeAttachment spec source.
`kube_volumeattachment_status_attached`	Gauge	Whether the VolumeAttachment is attached.
`kube_volumeattachment_status_attachment_metadata`	Gauge	Metadata of the attached volume.

Workload

Metrics related to overall resource consumption and performance of your applications. They track key aspects like CPU usage, memory utilization, and network traffic of your deployments, pods, and other workloads.

`CronJob` Metrics

Metric name	Metric type	Description
`kube_cronjob_annotations`	Gauge	Kubernetes annotations converted to Prometheus labels controlled via `--metric-annotations-allowlist` .
`kube_cronjob_info`	Gauge	Information about CronJobs (such as, schedule, concurrency policy, timezone).
`kube_cronjob_labels`	Gauge	Kubernetes labels converted to Prometheus labels controlled via `--metric-annotations-allowlist` .
`kube_cronjob_created`	Gauge	Creation timestamp (in Unix time) of the CronJob.

Metric name	Metric type	Description
kube_cronjob_next_schedule_time	Gauge	Time of the next scheduled execution of the CronJob.
kube_cronjob_status_active	Gauge	Whether the CronJob is actively running Jobs.
kube_cronjob_status_last_schedule_time	Gauge	Time of the last attempted execution of the CronJob.
kube_cronjob_status_last_successful_time	Gauge	Time of the last successful execution of the CronJob.
kube_cronjob_spec_suspend	Gauge	Whether the CronJob is suspended.
kube_cronjob_spec_starting_deadline_seconds	Gauge	Deadline in seconds for the CronJob to complete the execution of its Jobs.
kube_cronjob_metadata_resource_version	Gauge	Resource version of the CronJob object.
kube_cronjob_spec_successful_job_history_limit	Gauge	Maximum number of successful Jobs to retain.
kube_cronjob_spec_failed_job_history_limit	Gauge	Maximum number of failed Jobs to retain.

`DaemonSet` Metrics

Metric name	Metric type	Description
kube_daemonset_annotations	Gauge	Kubernetes annotations converted to Prometheus labels controlled via `--metric-annotations-allowlist` .
kube_daemonset_created	Gauge	Creation timestamp (in Unix time) of the DaemonSet.
kube_daemonset_status_current_number_scheduled	Gauge	Current number of scheduled pods by the DaemonSet.

Metric name	Metric type	Description
kube_daemonset_status_desired_number_scheduled	Gauge	Desired number of scheduled pods by the DaemonSet.
kube_daemonset_status_number_available	Gauge	Number of available pods (ready and running) managed by the DaemonSet.
kube_daemonset_status_number_misscheduled	Gauge	Number of pods that are running but are not scheduled to be running on the current node.
kube_daemonset_status_number_ready	Gauge	Number of pods that are ready and running.
kube_daemonset_status_number_unavailable	Gauge	Number of unavailable pods managed by the DaemonSet.
kube_daemonset_status_observed_generation	Gauge	Generation of the DaemonSet that was last observed by the controller.
kube_daemonset_status_updated_number_scheduled	Gauge	The number of pods that have been scheduled by the DaemonSet controller.
kube_daemonset_metadata_generation	Gauge	Generation of the resource as seen by the server.
kube_daemonset_labels	Gauge	Kubernetes labels converted to Prometheus labels controlled via `--metric-annotations-allowlist` .

`Deployment` Metrics

Metric name	Metric type	Description
kube_deployment_annotations	Gauge	Kubernetes annotations converted to Prometheus labels controlled via --metric-annotations-allowlist .
kube_deployment_status_replicas	Gauge	Current number of replicas of the deployment.
kube_deployment_status_replicas_available	Gauge	Number of available pods (ready and running) managed by the deployment.
kube_deployment_status_replicas_unavailable	Gauge	Number of unavailable pods managed by the deployment.
kube_deployment_status_replicas_updated	Gauge	The number of pods that have been updated by the deployment controller.
kube_deployment_status_observed_generation	Gauge	Generation of the deployment that was last observed by the controller.
kube_deployment_spec_replicas	Gauge	Desired number of replicas for the deployment.
kube_deployment_spec_paused	Gauge	Whether the deployment is paused.
kube_deployment_spec_strategy_rollingupdate_max_unavailable	Gauge	Maximum unavailable replicas during a rolling update.

Metric name	Metric type	Description
kube_deployment_spec_strategy_rollingupdate_max_surge	Gauge	Maximum surge replicas during a rolling update.
kube_deployment_metadata_generation	Gauge	Generation of the resource as seen by the server.
kube_deployment_created	Gauge	Creation timestamp (in Unix time) of the deployment.

`Horizontal Pod Autoscaler` Metrics

Metric name	Metric type	Description
kube_horizontalpodautoscaler_info	Gauge	Information about the Horizontal Pod Autoscaler (HPA).
kube_horizontalpodautoscaler_annotations	Gauge	Kubernetes annotations converted to Prometheus labels controlled via `--metric-annotations-allowlist` .
kube_horizontalpodautoscaler_labels	Gauge	Kubernetes labels converted to Prometheus labels controlled via `--metric-annotations-allowlist` .
kube_horizontalpodautoscaler_metadata_generation	Gauge	Generation of the HPA resource as seen by the server.
kube_horizontalpodautoscaler_spec_max_replicas	Gauge	Maximum number of replicas allowed for the target resource.
kube_horizontalpodautoscaler_spec_min_replicas	Gauge	Minimum number of replicas allowed for the target resource.

Metric name	Metric type	Description
kube_horizontalpodautoscaler_spec_target_metric	Gauge	Target metric for the HPA.
kube_horizontalpodautoscaler_status_target_metric	Gauge	Current value of the target metric.
kube_horizontalpodautoscaler_status_condition	Gauge	Condition of the HPA (such as, True, False, Unknown).
kube_horizontalpodautoscaler_status_current_replicas	Gauge	Current number of replicas of the target resource.
kube_horizontalpodautoscaler_status_desired_replicas	Gauge	Desired number of replicas for the target resource as determined by the HPA.

Job Metrics

Metric name	Metric type	Description
kube_job_annotations	Gauge	Kubernetes annotations converted to Prometheus labels controlled via `--metric-annotations-allowlist` .
kube_job_info	Gauge	Information about the Job.
kube_job_labels	Gauge	Kubernetes labels converted to Prometheus labels controlled via `--metric-annotations-allowlist`.
kube_job_owner	Gauge	Owner of the Job.
kube_job_spec_parallelism	Gauge	Parallelism of the Job.
kube_job_spec_completions	Gauge	Completions of the Job.
kube_job_spec_active_deadline_seconds	Gauge	Active deadline seconds of the Job.
kube_job_status_active	Gauge	Whether the Job is active.
kube_job_status_succeeded	Gauge	Whether the Job succeeded.

Metric name	Metric type	Description
kube_job_status_failed	Gauge	Whether the Job failed.
kube_job_status_start_time	Gauge	Start time of the Job.
kube_job_status_completion_time	Gauge	Completion time of the Job.
kube_job_complete	Gauge	Job completion condition.
kube_job_failed	Gauge	Job failed condition.
kube_job_created	Gauge	Job creation time.
kube_job_status_suspended	Gauge	Whether the Job is suspended.

Pod Metrics

Metric name	Metric type	Description
kube_pod_annotations	Gauge	Kubernetes annotations converted to Prometheus l controlled via --metric-annotations-allowlist
kube_pod_info	Gauge	Information about pod
kube_pod_ips	Gauge	Pod IP addresses
kube_pod_start_time	Gauge	Start time for a pod (times Unix time)
kube_pod_completion_time	Gauge	Completion time for a pod (timestamp in Unix time)
kube_pod_owner	Gauge	Information about the Pod owner
kube_pod_labels	Gauge	Kubernetes labels convert Prometheus labels control --metric-annotations-allowlist .
kube_pod_nodeselectors	Gauge	Describes the Pod nodeSe

Metric name	Metric type	Description
kube_pod_status_phase	Gauge	The pods current phase
kube_pod_status_qos_class	Gauge	The pods current qosClass
kube_pod_status_ready	Gauge	Describes whether the poc ready to serve requests
kube_pod_status_scheduled	Gauge	Describes the status of the scheduling process for the
kube_pod_container_info	Gauge	Information about a conta a pod
kube_pod_container_status_waiting	Gauge	Describes whether the cor is currently in waiting state
kube_pod_container_status_waiting_reason	Gauge	Describes the reason the container is currently in wa state
kube_pod_container_status_running	Gauge	Describes whether the cor is currently in running stat
kube_pod_container_state_started	Gauge	Start time for a pod contai (timestamp in Unix time)
kube_pod_container_status_terminated	Gauge	Describes whether the cor is currently in terminated s
kube_pod_container_status_terminated_reason	Gauge	Describes the reason the container is currently in terminated state
kube_pod_container_status_last_terminated_reason	Gauge	Describes the last reason t container was in terminate
kube_pod_container_status_last_terminated_exitcode	Gauge	Describes the exit code for last container in terminate
kube_pod_container_status_last_terminated_timestamp	Gauge	Last terminated time for a container (timestamp in Ur time).
kube_pod_container_status_ready	Gauge	Describes whether the cor readiness check succeeded
kube_pod_status_initialized_time	Gauge	Time when the pod is initia

Metric name	Metric type	Description
kube_pod_status_ready_time	Gauge	Time when pod passed rea probes.
kube_pod_status_container_ready_time	Gauge	Time when the container c pod entered Ready state.
kube_pod_container_status_restarts_total	Counter	The number of container r per container
kube_pod_container_resource_requests	Gauge	The number of requested resource by a container. It recommended to use the kube_pod_resource_re metric exposed by kube-scheduler instead, as it is r precise.
kube_pod_container_resource_limits	Gauge	The number of requested resource by a container. It recommended to use the kube_pod_resource_li metric exposed by kube-scheduler instead, as it is r precise.
kube_pod_overhead_cpu_cores	Gauge	The pod overhead in regar cpu cores associated with running a pod
kube_pod_overhead_memory_bytes	Gauge	The pod overhead in regar memory associated with ru a pod
kube_pod_runtimeclass_name_info	Gauge	The runtimeclass associate the pod
kube_pod_created	Gauge	Creation timestamp (in Un
kube_pod_deletion_timestamp	Gauge	Deletion timestamp (in Un
kube_pod_restart_policy	Gauge	Describes the restart polic by this pod
kube_pod_init_container_info	Gauge	Information about an init container in a pod
kube_pod_init_container_status_waiting	Gauge	Describes whether the init container is currently in wa

Metric name	Metric type	Description
		state
kube_pod_init_container_status_waiting_reason	Gauge	Describes the reason the i container is currently in wa state
kube_pod_init_container_status_running	Gauge	Describes whether the init container is currently in ru state
kube_pod_init_container_status_terminated	Gauge	Describes whether the init container is currently in terminated state
kube_pod_init_container_status_terminated_reason	Gauge	Describes the reason the i container is currently in terminated state
kube_pod_init_container_status_last_terminated_reason	Gauge	Describes the last reason t container was in terminate
kube_pod_init_container_status_ready	Gauge	Describes whether the init containers readiness checl succeeded
kube_pod_init_container_status_restarts_total	Counter	The number of restarts for init container
kube_pod_init_container_resource_limits	Gauge	The number of CPU cores requested limit by an init container
kube_pod_init_container_resource_requests	Gauge	The number of CPU cores requested by an init contai
kube_pod_spec_volumes_persistentvolumeclaims_info	Gauge	Information about persistentvolumeclaim vol in a pod
kube_pod_spec_volumes_persistentvolumeclaims_readonly	Gauge	Describes whether a persistentvolumeclaim is mounted read only
kube_pod_status_reason	Gauge	The pod status reasons
kube_pod_status_scheduled_time	Gauge	Timestamp (in Unix time) v pod moved into scheduled

Metric name	Metric type	Description
kube_pod_status_unschedulable	Gauge	Describes the unschedulab status for the pod
kube_pod_tolerations	Gauge	Information about the pod tolerations
kube_pod_service_account	Gauge	The service account for a p
kube_pod_scheduler	Gauge	The scheduler for a pod

ReplicaSet Metrics

Metric name	Metric type	Description
kube_replicaset_annotations	Gauge	Kubernetes annotations converted to Prometheus labels controlled via --metric-annotations-allowlist .
kube_replicaset_status_replicas	Gauge	Number of replicas in the current ReplicaSet.
kube_replicaset_status_fully_labeled_replicas	Gauge	Number of fully labeled replicas in the current ReplicaSet.
kube_replicaset_status_ready_replicas	Gauge	Number of ready replicas in the current ReplicaSet.
kube_replicaset_status_observed_generation	Gauge	Observed generation of the ReplicaSet.
kube_replicaset_spec_replicas	Gauge	Replicas is the number of desired replicas.
kube_replicaset_metadata_generation	Gauge	Generation of the ReplicaSet as seen by the server.
kube_replicaset_labels	Gauge	Kubernetes labels converted to Prometheus labels controlled via --metric-annotations-allowlist .
kube_replicaset_created	Gauge	Creation timestamp (in Unix time).

Metric name	Metric type	Description
kube_replicaset_owner	Gauge	Owner of the ReplicaSet.

`ReplicationController` Metrics

Metric name	Metric type	Description
		Number of replicas in

17.7. Linux System Metrics Details

Consult the reference for the metrics reported by the System Metrics Source on Linux.

Events generated by the System Metrics Source have metrics metadata to designate dimension and metric fields. The `host` field contains the hostname, and is included as a dimension in all of them. The collectors include:

- System
- CPU
- Memory
- Network
- Disk
- Container
- Process

In the Source's configuration modal, you can set the level of detail for each type of metrics:

- **Basic** enables minimal metrics, averaged or aggregated.
- **All** enables full, detailed metrics, specified for individual CPUs, interfaces, and so on.
- **Custom** displays sub-menus and buttons from which you can choose a level of detail (Basic, All, Custom, or Disabled) for each type of event.
- **Disabled** means that no metrics will be generated.

Basic and **Custom** have different meanings depending on event type and will be described under each section below.

The tables outline the metrics emitted for each mode (Basic or Custom) and where applicable, the dimensions (to indicate where the metrics are coming from).

System

With System Metrics enabled, Cribl Edge captures CPU load averages, uptime, and count. The **Custom** option allows you to include process metrics that reflect the numbers of processes in various states.

Metrics for the overall system include the following:

Name	Description	Type	Dimensions	Mode
`node_uname_info`	Labeled system information as provided by the uname system call.	Counter	`release`, `sysname`, `version`	Basic
`node_cpu_count`	The number of CPU cores.	Gauge	`release`, `sysname`, `version`	Basic
`node_uptime_seconds`	System uptime in seconds.	Counter	N/A	Basic
`node_boot_time_seconds`	Node boot time in Unix time.	Counter	N/A	Basic
`node_time_seconds`	System time in seconds.	Counter	N/A	Basic
`node_load1`	1m load average.	Gauge	N/A	Basic
`node_load5`	5m load average.	Gauge	N/A	Basic
`node_load15`	15m load average.	Gauge	N/A	Basic
`node_open_fds`	Open file descriptors	Counter	N/A	Basic
`node_processes_state_all`	Total number of processes in different states.	Gauge	`state`	Basic
`node_processes_threads`	Allocated threads in system.	Gauge	`state`	Basic
`node_procs_blocked`	Number of processes blocked waiting for I/O to complete.	Gauge	`state`	Basic
`node_procs_running`	Number of processes in runnable state.	Gauge	`state`	Basic
`node_processes_state`	Number of processes in each state.	Gauge	`state`	Custom: `Process metrics`

CPU

Cribl Edge captures active, user, system, idle, and `iowait` percentages over all CPUs, with options to add per-CPU metrics and raw time counters for each state.

Metrics for CPUs include the following:

Name	Description	Type	Dimensions	Mode
`node_cpu_percent_active_all`	Percent all the CPUs spent in activity.	Gauge	N/A	Basic
`node_cpu_seconds_active_all_total`	Seconds all the CPUs spent in activity (excluding idle and wait).	Counter	N/A	Custom: `CPU time metrics`
`node_cpu_seconds_active_total`	Seconds each CPU spent in activity (excluding idle and wait).	Counter	`cpu`	Custom: `CPU time metrics`
`node_cpu_seconds_all_total`	Seconds for all CPUs usage.	Counter	`mode`	Custom: `CPU time metrics`
`node_cpu_seconds_total`	Seconds for each CPU's usage.	Counter	`cpu`, `mode`	Custom: `CPU time metrics`
`node_cpu_percent_active`	Percent each CPU spent in activity.	Gauge	`cpu`	Custom: `Per CPU` or `Detailed` metrics
`node_cpu_percent_all`	Percent CPU usage for all.	Gauge	`mode`,`user`	Custom: `Detailed` metrics
`node_cpu_percent`	Percent CPU usage for each CPU.	Gauge	`cpu`,`mode`,`user`	Custom: `Per CPU` or `Detailed` metrics

- The `Per CPU` option adds metrics with the `cpu` dimension.
- The `Detailed metrics` option adds metrics with `mode` dimension set to: `irq`, `softirq`, `steal`, `guest`, `guest_nice`, and `nice`.

Memory

With System Metrics enabled, Cribl Edge captures memory metrics including total, used, available, `swap_free`, and `swap_total`, with the option to toggle all memory states.

Metrics for memory include the following:

Name	Description	Type	Dimensions	Mode
`node_memory_MemTotal_bytes`	Memory information field `MemTotal_bytes`.	Gauge	N/A	Basic
`node_memory_Used_bytes`	Used memory in bytes.	Gauge	N/A	Basic
`node_memory_Used_percent`	Percent used memory.	Gauge	N/A	Basic
`node_memory_MemAvailable_bytes`	Memory information field `MemAvailable_bytes`.	Gauge	N/A	Basic
`node_memory_MemAvailable_percent`	Percent memory available.	Gauge	N/A	Basic
`node_memory_SwapFree_bytes`	Memory information field `SwapFree_bytes`.	Gauge	N/A	Basic
`node_memory_SwapTotal_bytes`	Memory information field `SwapTotal_bytes`.	Gauge	N/A	Basic
`node_vmstat_oom_kill`	`/proc/vmstat` information field `oom_kill`.	Gauge	N/A	Basic
`node_vmstat_pgfault`	`/proc/vmstat` information field `pgfault`.	Gauge	N/A	Basic
`node_vmstat_pgmajfault`	`/proc/vmstat` information field `pgmajfault`.	Gauge	N/A	Basic
`node_vmstat_pgpgin`	`/proc/vmstat` information field `pgpgin`.	Gauge	N/A	Basic
`node_vmstat_pgpgout`	`/proc/vmstat` information field `pgpgout`.	Gauge	N/A	Basic
`node_vmstat_pswpin`	`/proc/vmstat` information field	Gauge	N/A	Basic

Name	Description	Type	Dimensions	Mode
	`pswpin`.			
`node_vmstat_pswpout`	`/proc/vmstat` information field `pswpout`.	Gauge	N/A	Basic
`node_memory_Active_bytes`	Memory information field `Active_bytes`.	Gauge	N/A	Custom: `Detailed` metrics
`node_memory_Buffers_bytes`	Memory information field `Buffers_bytes`.	Gauge	N/A	Custom: `Detailed` metrics
`node_memory_Cached_bytes`	Memory information field `Cached_bytes`.	Gauge	N/A	Custom: `Detailed` metrics
`node_memory_Dirty_bytes`	Memory information field `Dirty_bytes`	Gauge	N/A	Custom: `Detailed` metrics
`node_memory_MemFree_bytes`	Memory information field `MemFree_bytes`.	Gauge	N/A	Custom: `Detailed` metrics
`node_memory_SwapCached_bytes`	Memory information field `SwapCached_bytes`.	Gauge	N/A	Custom: `Detailed` metrics

Network

With System Metrics enabled, Cribl Edge captures bytes, packets, errors, and network connections over all interfaces. The **Custom** option allows you to filter interfaces, and to decide whether to select per-interface metrics and generate protocol metrics.

Metrics for networks include the following:

Name	Description	Type	Dimensions	Mo
`node_network_receive_bytes_all_total`	Network device statistic `receive_bytes`.	Counter	N/A	Bas
`node_network_receive_errs_all_total`	Network device statistic `receive_errs`.	Counter	N/A	Bas

Name	Description	Type	Dimensions	Mo
node_network_receive_packets_all_total	Network device statistic receive_packets.	Counter	N/A	Bas
node_network_transmit_bytes_all_total	Network device statistic transmit_bytes.	Counter	N/A	Bas
node_network_transmit_errs_all_total	Network device statistic transmit_errs.	Counter	N/A	Bas
node_network_transmit_packets_all_total	Network device statistic transmit_packets.	Counter	N/A	Bas
node_socket_tcp_established_total	TCP established connections.	Counter	N/A	Bas
node_network_receive_bytes_total	Network device statistic receive_bytes per interface.	Counter	device	Cus Per Int
node_network_receive_errs_total	Network device statistic receive_errs per interface.	Counter	device	Cus Per Int
node_network_receive_packets_total	Network device statistic receive_packets per interface.	Counter	device	Cus Per Int
node_network_transmit_bytes_total	Network device statistic transmit_bytes per interface.	Counter	device	Cus Per Int
node_network_transmit_errs_total	Network device statistic transmit_errs per interface.	Counter	device	Cus Per Int
node_network_transmit_packets_total	Network device statistic transmit_packets per interface.	Counter	device	Cus Per Int

Name	Description	Type	Dimensions	Mo
node_network_receive_drop_all_total	Network device statistic receive_drop.	Counter	N/A	Cus Det Met
node_network_receive_drop_total	Network device statistic receive_drop per interface.	Counter	device	Cus Det Met
node_network_transmit_drop_all_total	Network device statistic transmit_drop.	Counter	N/A	Cus Det Met
node_network_transmit_drop_total	Network device statistic transmit_drop per interface.	Counter	device	Cus Det Met
node_socket_tcp_syn_sent_total	TCP sent packets total.	Counter	N/A	Cus Det Met
node_socket_tcp_syn_recv_total	TCP received packets total.	Counter	N/A	Cus Det Met
node_socket_tcp_fin_wait1_total	Total connections waiting for termination request from remote TCP.	Counter	N/A	Cus Det Met
node_socket_tcp_fin_wait2_total	Active TCP connections to be shut down.	Counter	N/A	Cus Det Met
node_socket_tcp_time_wait_total	Length of time to pass to be sure the remote TCP received the acknowledgement to terminate.	Counter	N/A	Cus Det Met
node_socket_tcp_close_total	Total TCP sockets with closed connections.	Counter	N/A	Cus Det Met

Name	Description	Type	Dimensions	Mo
`node_socket_tcp_last_ack_total`	Total TCP sockets in state before the TCP connection is closed.	Counter	N/A	Cus Det Met
`node_socket_tcp_listen_total`	Total TCP sockets waiting for a connection request from any remote TCP/port.	Counter	N/A	Cus Det Met
`node_socket_tcp_closing_total`	Total TCP sockets waiting for connection termination request acknowledgement.	Counter	N/A	Cus Det Met
`node_socket_tcp_none_total`	Number of TCP sockets with no connections.	Counter	N/A	Cus Det Met
`node_socket_udp_total`	Number of UDP sockets in use.	Counter	N/A	Cus Det Met

Disk

With System Metrics enabled, Cribl Edge captures disk-used metrics - in percent, bytes read and written, and read and write operations - over all mounted disks. The **Custom** option allows you to filter devices, mountpoint, and filesystem type, and to decide whether to select per-device metrics and generate detailed metrics.

Metrics for Disk include the following:

Name	Description	Type	Dimensions
`node_disk_reads_completed_all_total`	Total number of reads completed successfully.	Counter	N/A
`node_disk_read_bytes_all_total`	Total number of bytes read successfully.	Counter	N/A
`node_disk_writes_completed_all_total`	Network device statistic `receive_packets`.	Counter	N/A

Name	Description	Type	Dimensions
node_disk_written_bytes_all_total	Total number of bytes written successfully.	Counter	N/A
node_filesystem_size_bytes_all	Filesystem size in bytes.	Gauge	N/A
node_filesystem_avail_bytes_all	Filesystem space available to non-root users in bytes.	Gauge	N/A
node_filesystem_used_bytes_all	Filesystem used space in bytes.	Gauge	N/A
node_filesystem_used_percent_all	Percent filesystem used space.	Gauge	N/A
node_filesystem_files_free_all	Filesystem free file nodes.	Gauge	N/A
node_filesystem_files_used_all	Filesystem total used file nodes.	Gauge	N/A
node_filesystem_files_used_percent_all	Percent Filesystem used in all disks.	Gauge	N/A
node_filesystem_size_bytes_all	Filesystem size in bytes.	Gauge	N/A
node_filesystem_used_bytes	Filesystem used space in bytes per disk.	Gauge	device, fstype, mountpoint
node_filesystem_used_percent	Percent Filesystem used per disk.	Gauge	device, fstype, mountpoint
node_disk_reads_completed_total	Total number of reads completed successfully per disk.	Counter	device
node_disk_read_bytes_total	Total number of bytes read successfully per disk.	Counter	device, fstype, mountpoint

Name	Description	Type	Dimensions
node_disk_writes_completed_total	Network device statistic receive_packets per disk.	Counter	device, fstype, mountpoint
node_disk_written_bytes_total	Total number of bytes written successfully per disk.	Counter	device, fstype, mountpoint
node_disk_discards_completed_all_total	Total number of discards completed successfully.	Counter	N/A
node_disk_discards_completed_total	Total number of discards completed successfully per disk.	Counter	device, fstype, mountpoint
node_disk_discards_merged_all_total	Total number of discards merged.	Counter	N/A
node_disk_read_time_seconds_total	Total number of seconds spent by all reads per disk.	Counter	device, fstype, mountpoint
node_disk_write_time_seconds_all_total	Total number of seconds spent by all writes.	Counter	N/A
node_disk_read_time_seconds_all_total	Total number of seconds spent by all reads.	Counter	N/A
node_disk_write_time_seconds_total	Total number of seconds spent by all writes per disk.	Counter	device, fstype, mountpoint
node_disk_reads_merged_all_total	Total number of reads merged.	Counter	N/A
node_disk_writes_merged_all_total	Total number of writes merged.	Counter	N/A

Name	Description	Type	Dimensions
node_disk_reads_merged_total	Total number of reads merged per disk.	Counter	device, fstype, mountpoint
node_disk_writes_merged_total	Total number of writes merged per disk.	Counter	device, fstype, mountpoint
node_disk_discards_merged_total	Total number of discards merged per disk.	Counter	device, fstype, mountpoint
node_disk_io_time_seconds_all_total	Total seconds spent doing I/Os.	Counter	N/A
node_disk_io_time_seconds_total	Total seconds spent doing I/Os per disk.	Counter	device, fstype, mountpoint
node_disk_io_now_all	The number of I/Os currently in progress.	Gauge	N/A
node_disk_io_now	The number of I/Os currently in progress per disk.	Counter	device, fstype, mountpoint
node_disk_io_time_weighted_seconds_all_total	Weighted number of seconds spent doing I/Os per device.	Counter	device, fstype, mountpoint
node_filesystem_size_bytes	Filesystem size in bytes per disk.	Gauge	device, fstype, mountpoint
node_filesystem_avail_bytes	Filesystem space available to non-root users in bytes per disk.	Gauge	device, fstype, mountpoint

Name	Description	Type	Dimensions
`node_filesystem_files_free`	Filesystem free file nodes per disk.	Gauge	`device`, `fstype`, `mountpoint`
`node_filesystem_files_used_percent`	Percent Filesystem used per disk.	Gauge	`device`, `fstype`, `mountpoint`
`node_filesystem_size_bytes`	Filesystem size in bytes per disk.	Gauge	`device`, `fstype`, `mountpoint`

Container

With System Metrics enabled, Cribl Edge generates Docker information with CPU, memory, network, and disk metrics for running containers. Optionally, you can customize which containers to generate metrics from.

Metrics for Container include the following:

Name	Description	Type
`container_start_time_seconds`	Unix time (seconds) when the container was started.	Counter
`container_finish_time_seconds`	Unix time (seconds) when the container was stopped. Only for non-running containers.	Counter
`container_fs_reads_bytes_all_total`	Total bytes read for all disk devices.	Counter
`container_memory_usage_percent`	Percent of available memory being used.	Gauge
`container_network_receive_bytes_all_total`	Total bytes received for all network interfaces.	Counter
`container_network_receive_errors_all_total`	Total number of errors received for all network interfaces.	Counter
`container_network_receive_packets_all_total`	Total number of packets received for all network interfaces.	Counter

Name	Description	Type
container_network_transmit_bytes_all_total	Total bytes transmitted for all network interfaces.	Counter
container_network_transmit_errors_all_total	Total number of errors transmitted for all network interfaces.	Counter
container_network_transmit_packets_all_total	Total number of packets transmitted for all network interfaces.	Counter
container_memory_total_bytes	Total number of memory bytes available.	Counter
container_cpu_user_seconds_total	Number of seconds the container has been on the CPU running user code.	Counter
container_cpu_system_seconds_total	Number of seconds the container has been on the CPU running kernel code.	Counter
container_fs_reads_bytes_total	Total bytes read per device	Counter
container_fs_writes_bytes_all_total	Total bytes written for all disk devices	Counter
container_fs_writes_bytes_total	Total bytes written per device	Counter

Name	Description	Type
container_fs_reads_all_total	Total number of read operations for all disk devices.	Counter
container_fs_writes_total	Total number write operations per device.	Counter
container_memory_mapped_file	Total bytes writted for all disk devices.	Counter
container_memory_max_usage_bytes	Highest seen value of the container_memory_usage_bytes metric.	Counter
container_memory_pgin	Total number of memory page-in events.	Counter
container_mem.pgpgout	Total number of memory pages paged out.	Counter
container_memory_pgfault	Total number of major page faults.	Counter
container_memory_pgmajfault	Total number of minor page faults.	Counter
container_memory_usage_bytes	Number of memory bytes used.	Counter
container_network_receive_dropped_all_total	Total number of receives dropped for all network interfaces.	Counter
container_network_transmit_dropped_all_total	Total number of transmits dropped for all network interfaces.	Counter

Process Metrics

With Process Metrics enabled, Cribl Edge captures process-specific metrics from Linux servers and reports them as events. This allows you to monitor specific processes on Cribl.Cloud instances. You can generate events for any process object.

For information on how to configure the System Metrics Source to generate process-specific metrics, check out the Process Metrics section of the System Metrics page.

Process-specific metrics are **not** affected by the **Host Metrics** detail setting.

Process-specific metrics include the following:

Name	Description	Type	Dimensions
`process_num_threads`	The number of threads.	Gauge	`process_cmdline`, `process_set`, `process_uid`, `process_gid`, `process_service`
`process_open_filedesc`	The number of file descriptors.	Gauge	`process_cmdline`, `process_set`, `process_uid`, `process_gid`, `process_service`
`process_write_bytes`	The number of bytes which this process caused to be sent to the storage layer.	Counter	`process_cmdline`, `process_set`, `process_uid`, `process_gid`, `process_service`
`process_read_bytes`	The number of bytes this process actually fetched from the storage layer. This number is accurate for block-backed filesystems.	Counter	`process_cmdline`, `process_set`, `process_uid`, `process_gid`, `process_service`
`process_major_page_faults`	The number of major faults for this process that required loading a memory page from disk.	Counter	`process_cmdline`, `process_set`, `process_uid`, `process_gid`, `process_service`

Name	Description	Type	Dimensions
process_minor_page_faults	The number of minor faults for this process that have not required loading a memory page from disk.	Counter	process_cmdline, process_set, process_uid, process_gid, process_service
process_voluntary_context_switches	The number of voluntary context switches.	Counter	process_cmdline, process_set, process_uid, process_gid, process_service
process_nonvoluntary_context_switches	The number of involuntary context switches.	Counter	process_cmdline, process_set, process_uid, process_gid, process_service
process_cpu_usage	The process's CPU usage, expressed as a percentage of total CPU power.	Gauge	process_cmdline, process_set, process_uid, process_gid, process_service
process_cpu_seconds	The process's CPU usage, based on user time and system time.	Counter	process_cmdline, process_set, process_uid, process_gid, process_service
process_resident_memory_bytes	The amount of memory used, in bytes. Includes the pages that count toward text, data, or stack space. Does not include pages that haven't been demand-loaded in or are swapped out.	Gauge	process_cmdline, process_set, process_uid, process_gid, process_service

17.8. Windows System Metrics Details

Consult the reference for the metrics reported by the System Metrics Source on Windows.

Events generated by the Windows Metrics Source include metrics metadata to designate dimension and metric fields. The `host` field contains the hostname, and is included as a dimension in all of them.

The collectors include:

- System
- CPU
- Memory
- Network
- Disk
- Process Metrics

In the Source's configuration modal, You can set the level of detail for each type of metrics:

- **Basic** enables minimal metrics, averaged or aggregated.
- **All** enables full, detailed metrics, specified for individual CPUs, interfaces, and so on.
- **Custom** displays sub-menus and buttons from which you can choose a level of detail (Basic, All, Custom, or Disabled) for each type of event.
- **Disabled** means that no metrics will be generated.

Basic and **Custom** have different meanings depending on event type and will be described under each section below.

The tables outline the metrics emitted for each mode (**Basic** or **Custom**) and where applicable, the dimensions (to indicate where the metrics are coming from).

System

With System Metrics enabled, Cribl Edge captures CPU load averages, uptime, and count. The **Custom** option option allows you to include detailed metrics. These are Windows-specific metrics including OS information, system uptime, CPU architecture, etc.

Metrics for the overall system include the following:

Name	Description	Type	Dimensions	Mode
windows_cs_logical_processors	Number of installed logical processors.	Gauge	N/A	Basic
windows_cs_physical_memory_bytes	Total installed physical memory.	Gauge	N/A	Basic
windows_os_info	Contains full product name & version in labels.	Gauge	product, version	Basic
windows_os_physical_memory_free_bytes	Bytes of physical memory currently unused and available.	Gauge	N/A	Basic
windows_os_processes	Number of process contexts currently loaded or running on the operating system.	Gauge	N/A	Basic
windows_system_processor_queue_length	Number of threads in the processor queue.	Gauge	N/A	Basic
windows_system_threads	Number of Windows system threads.	Gauge	N/A	Basic
windows_cs_hostname	Labeled system hostname information.	Gauge	hostname, domain, fqdn	Custo Deta

Name	Description	Type	Dimensions	Mode
windows_cpu_info	Labeled CPU information.	Gauge	architecture, device_id, description, family, l2_cache_size, l3_cache_size, name	Custo Deta
windows_os_paging_limit_bytes	Total number of bytes that can be stored in the operating system paging files.	Gauge	N/A	Custo Deta
windows_os_paging_free_bytes	Number of bytes that can be mapped into the operating system paging files without causing any other pages to be swapped out.	Gauge	N/A	Custo Deta
windows_os_processes_limit	Maximum number of process contexts the operating system can support.	Gauge	N/A	Custo Deta
windows_os_process_memory_limit_bytes	Maximum number of bytes of memory that can be allocated to a process.	Gauge	N/A	Custo Deta

Name	Description	Type	Dimensions	Mode
`windows_os_virtual_memory_bytes`	Bytes of virtual memory.	Gauge	N/A	Custo Deta
`windows_system_exception_dispatches_total`	Total exceptions dispatched by the system.	Counter	N/A	Custo Deta
`windows_system_system_calls_total`	Total combined calls to Windows NT system service routines by all processes running on the computer.	Counter	N/A	Custo Deta
`windows_system_system_up_time`	Time of last boot of system.	Gauge	N/A	Custo Deta

CPU

Basic level captures active, user, system, idle, and `iowait` percentages over all CPUs.

Custom level toggles the following on or off: **Per CPU metrics**, **Detailed metrics** (i.e., metrics for all CPU states), and **CPU time metrics** (i.e., raw, monotonic CPU time counters).

Metrics for CPUs include the following:

Name	Description	Type	Dimensions	Mode
`windows_cpu_percent_active_all`	CPU percent active usage	Gauge	`core`, `mode`	Basic
`windows_cpu_percent_active`	CPU percent active usage per CPU	Gauge	`core`, `mode`	Basic Custo Per and C time metri

Name	Description	Type	Dimensions	Mode
windows_cpu_percent	CPU percent active usage	Gauge	core, mode is set to user, idle, privileged, interrupt, dpc	Basic Custo Per and C time metri
windows_cpu_parking_status	Parking Status represents whether a processor is parked or not.	Counter	core	Basic Custo Per and C time metri
windows_cpu_core_frequency_mhz	Core frequency in megahertz.	Gauge	core	Basic Custo Per and C time metri
windows_cpu_time_all_total	Sum of all cpu_time across all cores.	Gauge	mode	Basic Custo CPU metri
windows_cpu_cstate_seconds_total	Time spent in low-power idle state.	Counter	core, state	Custo Per and Deta metri
windows_cpu_time_total	Time that processor spent in different modes (idle, user, system etc.).	Counter	core, mode	Custo Per and C time metri
windows_cpu_interrupts_total	Total number of received and serviced hardware interrupts.	Counter	core	Custo Per or Deta metri
windows_cpu_dpcs_total	Total number of	Counter	core	Custo Per

Name	Description	Type	Dimensions	Mode
	received and serviced deferred procedure calls (DPCs).			or Deta metri
windows_cpu_clock_interrupts_total	Total number of received and serviced clock tick interrupts.	Counter	core	Custo Per or Deta metri
windows_cpu_idle_break_events_total	Total number of time processor was woken from idle.	Counter	core	Custo Per or Deta metri
windows_cpu_processor_performance	Average performance of the processor while it is executing instructions.	Gauge.	core	Custo Per and Deta metri
windows_cpu_percent_processor_performance	Average performance of the processor while it is executing instructions, as a percentage of the nominal performance of the processor.	Gauge.	core	Custo Per and Deta metri
windows_cpu_percent_processor_utility	Amount of work a processor is completing, as a percentage of the amount of	Gauge.	core	Custo Per and Deta metri

Name	Description	Type	Dimensions	Mode
	work the processor could complete if it were running at its nominal performance and never idle.			
windows_cpu_average_idle_time	Processor idle time.	Gauge	mode	Custo Per and Deta metri
windows_cpu_percent_privilege_utility	Amount of work a processor is completing while executing in privileged mode.	Gauge	mode	Custo Per and Deta metri
windows_cpu_interrupts_total_per_sec	Total number of received and serviced hardware interrupts, computed average on a per second interval.	Gauge	mode	Custo Per and Deta metri
windows_cpu_dpcs_total_per_sec	Total number of received and serviced deferred procedure calls (DPCs), computed average on a per second interval.	Gauge	mode	Custo Per and Deta metri
windows_cpu_clock_interrupts_total_per_sec	Total number of	Gauge	mode	Custo Per

Name	Description	Type	Dimensions	Mode
	received and serviced clock tick interrupts, computed average on a per second interval.			and Deta metri
windows_cpu_idle_break_events_total_per_sec	Total number of time processor was woken from idle, computed average on a per second interval.	Gauge	mode	Custc Per and Deta metri
windows_cpu_percent_all_total	Percentage of all cpu_time across all cores.	Gauge	mode is set to dpc, idle, interrupt, privilege, user,active	Custc CPU metri

Memory

Basic level captures captures total, used, available, `swap_free`, and `swap_total`.

Custom level toggles **Detailed metrics** on or off. (These are metrics for all memory states.)

Metrics for memory include the following:

Name	Description	Type	Dimensic
windows_memory_available_bytes	Physical memory that is immediately available for allocation to a process or for system use. This is the sum of the standby (cached), free, and zero page lists.	Gauge	N/A

Name	Description	Type	Dimensio
windows_memory_cache_bytes	Number of bytes currently being used by the filesystem cache	Gauge	N/A
windows_memory_cache_bytes_peak	Maximum number of CacheBytes after the system was last restarted.	Gauge	N/A
windows_memory_cache_faults_total	Faults that occur when a page sought in the filesystem cache is not found there and must be retrieved elsewhere in memory (soft fault) or from disk (hard fault).	Counter	N/A
windows_memory_commit_limit	Bytes of virtual memory that can be committed without having to extend paging files.	Gauge	N/A
windows_memory_committed_bytes	Bytes of committed virtual memory.	Gauge	N/A
windows_memory_pool_paged_allocs_total	Calls to allocate space in the paged pool, regardless of the amount of space allocated in each call.	Counter	N/A
windows_memory_pool_paged_bytes	Number of bytes in the paged pool.	Gauge	N/A
windows_memory_pool_paged_resident_bytes	The size, in bytes, of the	Gauge	N/A

Name	Description	Type	Dimensio
	portion of the paged pool that is currently resident and active in physical memory. The paged pool is an area of the system virtual memory used for objects that can be written to disk when they are not being used.		
windows_memory_demand_zero_faults_total	Number of Zeroed pages required to satisfy faults. Windows uses zeroed pages as a security measure to prevent processes from seeing data stored by earlier processes that previously used the memory space.	Counter	N/A
windows_memory_free_and_zero_page_list_bytes	Physical memory allocated to free and zero pages, in bytes. This memory does not contain cached data. It is immediately available for allocation to a process or system use.	Gauge	N/A
windows_memory_free_system_page_table_entries	Page table entries not being used by the system.	Gauge	N/A

Name	Description	Type	Dimensio
windows_memory_modified_page_list_bytes	Physical memory, in bytes, assigned to the modified page list. This memory contains cached data and code that is not actively in use by processes, the system, and the system cache. This memory needs to be written out before it will be available for allocation to a process or for system use.	Gauge	N/A
windows_memory_page_faults_total	Overall rate at which faulted pages are handled by the processor.	Counter	N/A
windows_memory_swap_page_reads_total	Disk page reads (a single read operation reading several pages is still only counted once).	Counter	N/A
windows_memory_swap_pages_read_total	Pages read across all page reads (i.e., counting all pages read even if they are read in a single operation).	Counter	N/A
windows_memory_swap_pages_written_total	Pages written across all page writes (i.e., counting all pages written even if they are written in a	Counter	N/A

Name	Description	Type	Dimensio
	single operation).		
windows_memory_swap_page_operations_total	Total number of swap page read and writes (PagesPersec).	Counter	N/A
windows_memory_swap_page_writes_total	Disk page writes (a single write operation writing several pages is still only counted once).	Counter	N/A
windows_memory_pool_nonpaged_allocs_total	The number of calls to allocate space in the non-paged pool. The non-paged pool is an area of system memory area for objects that cannot be written to disk, and must remain in physical memory as long as they are allocated	Counter.	N/A
windows_memory_pool_nonpaged_bytes	Non-paged pool, in bytes. The non-paged pool is an area of the system virtual memory that is used for objects that cannot be written to disk, but must remain in physical memory as long as they are allocated.	Gauge	N/A
windows_memory_standby_cache_core_bytes	Physical memory, in bytes, that is assigned to the	Gauge	N/A

Name	Description	Type	Dimensio
	core standby cache page lists. This memory contains cached data and code that is not actively in use by processes, the system, and the system cache. It is immediately available for allocation to a process or for system use. If the system runs out of available free and zero memory, memory on lower priority standby cache page lists will be repurposed before memory on higher priority standby cache page lists.		
`windows_memory_standby_cache_normal_priority_bytes`	Physical memory, in bytes, that is assigned to the normal priority standby cache page lists. This memory contains cached data and code that is not actively in use by processes, the system, and the system cache. It is immediately available for allocation to a process or for system use. If the system runs out of available free and zero	Gauge	N/A

Name	Description	Type	Dimensio
	memory, memory on lower priority standby cache page lists will be repurposed before memory on higher priority standby cache page lists.		
windows_memory_standby_cache_reserve_bytes	Pysical memory, in bytes, that is assigned to the reserve standby cache page lists. This memory contains cached data and code that is not actively in use by processes, the system and the system cache. It is immediately available for allocation to a process or for system use. If the system runs out of available free and zero memory, memory on lower priority standby cache page lists will be repurposed before memory on higher priority standby cache page lists.	Gauge	N/A

Name	Description	Type	Dimensic
windows_memory_system_cache_resident_bytes	The size, in bytes, of the portion of the system file cache which is currently resident and active in physical memory.	Gauge	N/A
windows_memory_system_code_resident_bytes	The size, in bytes, of the pageable operating system code that is currently resident and active in physical memory. This value is a component of Memory/System Code Total Bytes. Memory/System Code Resident Bytes (and Memory/System Code Total Bytes) does not include code that must remain in physical memory and cannot be written to disk.	Gauge	N/A
windows_memory_system_code_total_bytes	The size, in bytes, of the pageable operating system code currently mapped into the system virtual address space. This value is calculated by summing the bytes in Ntoskrnl.exe,	Gauge	N/A

Name	Description	Type	Dimensio
	Hal.dll, the boot drivers, and filesystems loaded by Ntldr/osloader. This counter does not include code that must remain in physical memory and cannot be written to disk.		
windows_memory_system_driver_resident_bytes	The size, in bytes, of the pageable physical memory being used by device drivers. It is the working set (physical memory area) of the drivers. This value is a component of Memory/System Driver Total Bytes, which also includes driver memory that has been written to disk. Neither Memory/System Driver Resident Bytes nor Memory/System Driver Total Bytes includes memory that cannot be written to disk.	Gauge	N/A
windows_memory_system_driver_total_bytes	The size, in bytes, of the pageable virtual memory currently being used by device drivers. Pageable memory can be	Gauge	N/A

Name	Description	Type	Dimensio
	written to disk when it is not being used. It includes both physical memory (Memory/System Driver Resident Bytes) and code and data paged to disk. It is a component of Memory/System Code Total Bytes.		
windows_memory_transition_faults_total	Rate at which page faults are resolved, by recovering pages that were being used by another process sharing the page, or were on the modified page list or the standby list, or were being written to disk at the time of the page fault. The pages were recovered without additional disk activity. Transition faults are counted in numbers of faults; because only one page is faulted in each operation, it is also equal to the number of pages faulted.	Counter	N/A
windows_memory_transition_pages_repurposed_total	Rate at which the number of transition cache pages were reused for a	Counter	N/A

Name	Description	Type	Dimensio
	different purpose. These pages would have otherwise remained in the page cache to provide a (fast) soft fault (instead of retrieving it from backing store) in the event the page was accessed in the future.		
windows_memory_write_copies_total	The number of page faults caused by attempting to write that were satisfied by copying the page from elsewhere in physical memory.	Counter	N/A
windows_memory_used_percent	Percent of committed memory used.	Gauge	N/A
windows_memory_available_percent	Percent of committed memory available.	Gauge	N/A
windows_memory_cache_faults_per_sec	Rate of cache faults computed per sec.	Gauge	N/A
windows_memory_demand_zero_faults_per_sec	Rate of Zeroed pages required to satisfy faults computed per sec.	Gauge	N/A
windows_memory_page_faults_per_sec	Rate at which faulted pages are handled by the processor	Gauge	N/A

Name	Description	Type	Dimensio
	computed per sec.		
windows_memory_page_reads_per_sec	Disk page reads computed per sec.	Gauge	N/A
windows_memory_pages_input_per_sec	Disk page reads computed per sec.	Gauge	N/A
windows_memory_pages_output_per_sec	Pages written across all page writes computed per sec.	Gauge	N/A
windows_memory_pages_per_sec	Total number of swap page read and writes computed per sec.	Gauge	N/A
windows_memory_page_writes_per_sec	Disk page writes computed per sec.	Gauge	N/A
windows_memory_transition_faults_sec	Rate at which page faults are resolved computed per sec.	Gauge	N/A
windows_memory_transition_pages_repurposed_per_sec	Rate at which the number of transition cache pages were reused for a different purpose computed per sec.	Gauge	N/A
windows_memory_write_copies_per_sec	Rate at which the number of page faults caused by attempting to write that were satisfied by copying the page from elsewhere	Gauge	N/A

Name	Description	Type	Dimensio
	in physical memory computed per sec.		

Network

Basic level captures bytes, packets, errors, and connections over all interfaces.

Custom level exposes the following:

- The **Interface filter**, which specifies which network interfaces to include or exclude. (An empty filter will include all metrics.)
- **Per interface metrics**, which toggle on or off.
- **Detailed metrics**, which toggle on or off. If on, the **Protocol metrics** toggle appears, allowing you to choose whether to generate metrics for ICMP, ICMPMsg, IP, TCP, UDP, and UDPLite.

Metrics for networks include the following:

Name	Description	Type	Dimen
`windows_net_packets_outbound_discarded_total`	Total outbound packets to be discarded even though no errors had been detected to prevent transmission.	Counter	`nic`
`windows_net_packets_outbound_errors_total`	Total packets that could not be transmitted due to errors.	Counter	`nic`
`windows_net_packets_received_discarded_total`	Total inbound packets that were chosen to be discarded even though no errors had been	Counter	`nic`

Name	Description	Type	Dimen
	detected to prevent delivery.		
windows_net_packets_received_errors_total	Total packets that could not be received due to errors.	Counter	nic
windows_net_packets_received_unknown_total	Total packets received by interface that were discarded because of an unknown or unsupported protocol.	Counter	nic
windows_net_packets_received_non_unicast_total	Total non-unicast (subnet broadcast or subnet multicast) packets that are delivered to a higher-layer protocol.	Counter	nic
windows_net_packets_received_non_unicast_total_per_sec	Rate at which non-unicast (subnet broadcast or subnet multicast) packets are delivered to a higher-layer protocol computed per sec.	Gauge	nic
windows_net_packets_received_unicast_total	Total subnet-unicast packets that are delivered to a higher-	Counter	nic

Name	Description	Type	Dimen
	layer protocol.		
windows_net_packets_received_unicast_total_per_sec	Rate at which subnet-unicast packets are delivered to a higher-layer protocol computed per sec.	Gauge	nic
windows_net_packets_sent_unicast_total	Total packets requested to be transmitted to subnet-unicast addresses by higher-level protocols.	Counter	nic
windows_net_packets_sent_non_unicast_total_per_sec	Total packets that are requested to be transmitted to nonunicast (subnet broadcast or subnet multicast) addresses by higher-level protocols.	Gauge	nic
windows_net_bytes_received_total	Total bytes received by interface.	Counter	nic
windows_net_bytes_received_total_per_sec	Total bytes received by interface computed per sec.	Gauge	nic
windows_net_bytes_sent_total	Total bytes transmitted	Counter	nic

Name	Description	Type	Dimen
	by interface.		
windows_net_bytes_sent_total_per_sec	Total bytes transmitted by interface computed per sec.	Gauge	nic
windows_net_bytes_total	Total bytes received and transmitted by interface.	Counter	nic
windows_net_bytes_total_per_sec	Total bytes received and transmitted by interface per sec.	Gauge	nic
windows_net_packets_received_total_per_sec	Total packets received by interface computed per sec.	Counter	nic
windows_net_bytes_total_per_sec	Total bytes received and transmitted by interface per sec.	Gauge	nic
windows_net_packets_total	Total packets received and transmitted by interface.	Counter	nic
windows_net_packets_total_per_sec	Total packets received and transmitted by interface computed per sec.	Gauge	nic
windows_net_packets_sent_total	Total packets transmitted	Counter	nic

Name	Description	Type	Dimen
	by interface.		
windows_net_packets_sent_total_per_sec	Total packets transmitted by interface computed per sec.	Gauge	nic
windows_net_packets_sent_unicast_total_per_sec	Rate at which packets are requested to be transmitted to subnet-unicast addresses by higher-level protocols computed per sec.	Gauge	nic
windows_net_packets_sent_non_unicast_total	Rate at which packets that are requested to be transmitted to nonunicast (subnet broadcast or subnet multicast) addresses by higher-level protocols per sec.	Counter	nic
windows_net_current_bandwidth_bytes	Estimate of the interface's current bandwidth in bytes per second.	Gauge	nic
windows_tcp_connection_failures_all_total	Number of times TCP connections have made a	Counter	af

Name	Description	Type	Dimen
	direct transition to the CLOSED state from the SYN-SENT state or the SYN-RCVD state, plus the number of times TCP connections have made a direct transition from the SYN-RCVD state to the LISTEN state.		
windows_tcp_connections_active_all_total	Number of times TCP connections have made a direct transition from the CLOSED state to the SYN-SENT state.	Counter	af
windows_tcp_connections_established	Number of TCP connections for which the current state is either ESTABLISHED or CLOSE-WAIT.	Gauge	af
windows_tcp_connections_passive_all_total	Number of times TCP connections have made a direct transition from the LISTEN state to the SYN-RCVD state.	Counter	af

Name	Description	Type	Dimen
windows_tcp_connections_reset_total	Number of times TCP connections have made a direct transition from the LISTEN state to the SYN-RCVD state.	Counter	af
windows_tcp_segments_total	Total segments sent or received using the TCP protocol.	Counter	af
windows_tcp_segments_received_all_total	Total segments received using the TCP protocol.	Counter	af
windows_tcp_segments_retransmitted_all_total	Total segments retransmitted using the TCP protocol.	Counter	af
windows_tcp_segments_sent_all_total	Total segments sent using the TCP protocol.	Counter	af
windows_tcp_segments_all_total_per_sec	Total segments sent or	Gauge	af

Name	Description	Type	Dimen
	received using the TCP protocol computed per sec.		
windows_tcp_segments_received_all_total_per_sec	Total segments received using the TCP protocol computed per sec.	Gauge	af
windows_tcp_segments_retransmitted_all_total_per_sec	Total segments retransmitted using the TCP protocol computed per sec.	Gauge	af
windows_tcp_segments_sent_all_total_per_sec	Total segments sent using the TCP protocol computed per sec.	Gauge	af
windows_net_datagrams_all_total	Total datagrams sent or received using the UDP protocol.	Counter	af

Name	Description	Type	Dimen
windows_net_datagrams_no_port_all_total	Rate of received UDP datagrams for which there was no application at the destination port.	Gauge	af
windows_net_datagrams_received_all_total	Rate at which UDP datagrams are delivered to UDP users.	Gauge	af
windows_net_datagrams_received_errors_all_total	Number of received UDP datagrams that could not be delivered excluding errors due to lack of an application at the destination port.	Gauge	af
windows_net_datagrams_sent_all_total	Total UDP datagrams sent from the entity.	Gauge	af
windows_net_datagrams_no_port_all_total_per_sec	Rate of received UDP datagrams for which there was no application at the destination	Gauge	af

Name	Description	Type	Dimen
	port computed per sec		
`windows_net_datagrams_received_all_total_per_sec`	Rate at which UDP datagrams are delivered to UDP users computed per sec	Gauge	`af`
`windows_net_datagrams_received_errors_all_total_per_sec`	Rate at which received UDP datagrams that could not be delivered, excluding errors due to lack of an application at the destination port, computed per sec.	Gauge	`af`
`windows_net_datagrams_sent_all_total_per_sec`	Rate at which UDP datagrams sent from the entity computed per sec.	Gauge	`af`

Disk

Basic level captures disk usage (%), bytes read and written, and read and write operations, over all mounted disks.

Custom level exposes the following:

- The **Volume filter**, specifying which Windows volumes to include or exclude. Supports wildcards and `!` (not) operators. An empty filter will include all volumes.
- **Per volume metrics**, which toggle on or off.

- **Detailed metrics**, which toggle on or off.

Metrics for Disk include the following:

Name	Description	Type	Dimensio
`windows_logical_disk_requests_queued`	Outstanding requests on the disk at the time the performance data is collected	Gauge	`volume`
`windows_logical_disk_read_bytes_total`	Rate at which bytes are transferred from the disk during read operations.	Counter	`volume`
`windows_logical_disk_reads_total`	Rate of read operations on the disk.	Counter	`volume`
`windows_logical_disk_write_bytes_total`	Rate at which bytes are transferred to the disk during write operations.	Counter	`volume`
`windows_logical_disk_writes_total`	Rate of write operations on the disk.	Counter	`volume`
`windows_logical_disk_write_latency_seconds_total`	Shows the average time, in seconds, of a write operation to the disk.	Counter	`volume`
`windows_logical_disk_read_latency_seconds_total`	Shows the average time, in seconds, of a read	Counter	`volume`

Name	Description	Type	Dimensio
	operation from the disk.		
windows_logical_disk_read_write_latency_seconds_total	Shows the time, in seconds, of the average disk transfer.	Counter	volume
windows_logical_disk_read_seconds_total	Seconds the disk was busy servicing read request.	Counter	volume
windows_logical_disk_idle_seconds_total	Seconds the disk was idle (not servicing read/write requests).	Counter	volume
windows_logical_disk_split_ios_total	Number of I/Os to the disk split into multiple I/Os.	Counter	volume
windows_logical_disk_percent_read_time	Percent rate of read operations on the disk.	Gauge	volume
windows_logical_disk_percent_write_time	Percent write operations on the disk.	Gauge	volume
windows_logical_disk_percent_time	Percent time the disk was in read + write operations	Gauge	volume
windows_logical_disk_percent_idle_time	Percent time the disk was idle.	Gauge	volume

Name	Description	Type	Dimensio
windows_logical_disk_percent_free_space	Percent space free on volume.	Gauge	volume
windows_logical_disk_average_disk_sec_per_transfer	Measures the average time of data reads and writes on the disk.	Gauge	volume
windows_logical_disk_average_disk_sec_per_read	Measures the average rate of disk read requests that are executed per second on a specific logical disk.	Gauge	volume
windows_logical_disk_average_disk_sec_per_write	Indicates how fast data is being written on average for a specific logical disk.	Gauge	volume
windows_logical_disk_split_ios_per_sec	Rate the I/Os to the disk were split into multiple I/Os per sec.	Gauge	volume
windows_logical_disk_bytes_per_sec	Exposes the rate bytes are transferred to or from the disk during write or read operations per sec.	Gauge	volume
windows_logical_disk_read_bytes_per_sec	Exposes the rate bytes are	Gauge	volume

Name	Description	Type	Dimensio
	transferred to or from the disk during read operations per sec.		
`windows_logical_disk_reads_per_sec`	Exposes the rate of read operations on the disk per sec.	Gauge	`volume`
`windows_logical_disk_transfers_per_sec`	How fast data is being read and written for a specific logical disk per sec.	Gauge	`volume`
`windows_logical_disk_write_bytes_per_sec`	Exposes the rate at which bytes are transferred from the disk during write operations per sec.	Gauge	`volume`

The `windows_logical_disk_free_bytes` and `windows_logical_disk_size_bytes` metrics are not updated in real time and might have a delay of 10-15min. This is the same behavior as the Windows performance counters.

Process Metrics

With Process Metrics enabled, Cribl Edge captures process-specific metrics from Windows servers and reports them as events. This allows you to monitor specific processes on Cribl.Cloud instances. You can generate events for any process object.

Process-specific metrics are **not** affected by the **Host Metrics** detail setting.

For information on how to configure the Windows Metrics Source to generate process-specific metrics, check out the section of the Windows Metrics page.

Process-specific metrics include the following:

Name	Description	Type	Dimensions
process_start_time	Time the process started.	gauge	process_cmdline, process_service
process_cpu_time_total	Elapsed time that the process's threads have spent executing instructions in either privileged mode or user mode. Included in this count is code executed to handle some hardware interrupts and trap conditions.	counter	process_cmdline, process_service
process_handles	Total number of handles the process has open. This number is the sum of the handles currently open by each thread in the process.	gauge	process_cmdline, process_service
process_io_bytes_total	Total number of bytes issued to I/O operations in either read, write, or other mode. This property counts all I/O activity generated by the process to include file, network, and device I/Os. Read and write modes include data operations; other mode includes those that don't involve data, like control operations.	counter	process_cmdline, process_service
process_io_operations_total	Total number of I/O operations issued in either read, write, or other mode. This property counts all I/O activity generated by the process to include file, network, and device I/Os. Read and write mode includes data operations; other mode includes those that do not involve data, such as control operations.	counter	process_cmdline, process_service
process_page_faults_total	Total number of page faults by threads executing in this process. A page fault	counter	process_cmdline, process_service

Name	Description	Type	Dimensions
	occurs when a thread refers to a virtual memory page that is not in its working set in main memory. This can cause the page not to be fetched from disk if it is on the standby list and hence already in main memory, or if it is in use by another process with which the page is shared.		
`process_page_file_bytes`	Current number of bytes this process has used in the paging files. Paging files are used to store pages of memory used by the process that are not contained in other files. Paging files are shared by all processes, and lack of space in paging files can prevent other processes from allocating memory.	gauge	`process_cmdline`, `process_service`
`process_pool_bytes`	Last observed number of bytes in the paged or nonpaged pool. The paged pool is an area of system memory (physical memory used by the operating system) for objects that can be written to disk when they are not being used. The nonpaged pool is an area of system memory (physical memory used by the operating system) for objects that cannot be written to disk, but must remain in physical memory as long as they are allocated. Nonpaged pool bytes are calculated differently than paged pool bytes, so they may not equal the total of paged pool bytes.	gauge	`process_cmdline`, `process_service`

Name	Description	Type	Dimensions

18. Troubleshoot Cribl Edge

Use the resources in this section to diagnose and resolve issues in Cribl Edge:

- Known Issues: Get information about issues that affect Cribl Edge, along with temporary workarounds and fix versions when applicable.
- Common Challenges on the Edge: Learn how to resolve common errors and warning messages in Cribl Edge.
- Diagnosing Issues: Create and export diagnostic bundles to share with Cribl Support.

18.1. Common Challenges On The Edge

Learn how to resolve common issues you can encounter in Cribl Edge.

Here are some common issues that users have encountered when using Cribl Edge, along with guidance to resolve them.

Edge Configuration Not Working

If your Cribl Edge configuration changes aren't taking effect, make sure you have committed and deployed them. It is an easy to forget step that is necessary for most configuration changes to take effect.

No Workers Registered

The "No workers registered" error can occur during a live data capture, when there are no Edge Nodes available to fulfill the capture request. This error can be triggered if you attempt to capture events too quickly following a commit & deploy.

You can do two things to resolve it:

- If you recently committed and deployed changes, wait 30 seconds and retry the live capture.
- In the Leader Node's sidebar, select **Edge Nodes** and confirm there are Nodes currently registered with the Leader. Note that the error appears on a per-Fleet basis, which means that even if you have registered Nodes, they may not be in the Fleet you're capturing live data from.

Duplicate Edge Node GUID

When you install and first run the software, a GUID is generated and stored in a `.dat` file located in `CRIBL_HOME/local/cribl/auth/` (For example: `/opt/cribl/local/cribl/auth/676f6174733432.dat` on Linux, or `C:\ProgramData\Cribl\local\cribl\auth\676f6174733432.dat` on Windows).

When you deploy Cribl Edge as part of a host image or VM, the presence of the `.dat` file can cause duplicate GUIDs. To avoid it, stop Cribl Edge, then remove the `.dat` file. The file will regenerate on the next run.

Other Common Challenges

To learn about other challenges that can occur when running specific Edge scenarios, consult:

- Windows Troubleshooting

- Windows Installation Troubleshooting
- Kubernetes Troubleshooting
- File Monitor Troubleshooting

18.2. Diagnose Issues

To help diagnose Cribl Edge problems, you can share a diagnostic bundle with Cribl Support. The bundle contains a snapshot of configuration files and logs at the time the bundle was created, and gives troubleshooters insights into how Cribl Edge was configured and operating at that time.

What's in the Diagnostic Bundle

The bundle includes the following subdirectories (and their contents) of `$CRIBL_HOME`:

- `.../default/*`
- `.../local/*` - except for `/local/cribl/auth/`, to exclude sensitive files.
- `.../log/*`
- `.../groups/*`
- `.../state/jobs/*` - will return all jobs if left empty.

As a security measure to protect sensitive information, the bundle excludes sample files, lookup files, and all certificate files (`.crt`, `.pem`, `.cer`, and `.key`) from all `$CRIBL_HOME` subdirectories.

Create and Export a Diagnostic Bundle

If you're managing your own Cribl Edge deployment (single-instance Stream/Edge or distributed Stream/Edge), you can create and securely share bundles with Cribl Support, either from the UI or via the CLI.

With a Cribl.Cloud deployment, create bundles and securely share them with Cribl Support directly from the UI. In the Using the UI instructions below, **Send to Cribl Support** will always be enabled, and the **Export**/download option will be unavailable.

Requirements

Using either medium, you'll need outbound internet access to `https://diag-upload.cribl.io` and a valid support case number. The above URL works only when using the `cribl diag` command or uploading using the Cribl Edge UI. (So connecting directly to it with your web browser will fail.)



If your organization doesn't permit outbound access to `https://diag-upload.cribl.io` to upload from within Cribl, you can also submit diags directly via the Support Portal (login required). If none of

> these options work with your organization's policies, please work directly with your Support engineer to find a solution.

Open your support case before sending a diagnostic bundle to Cribl Support. Then, when creating the bundle, be sure to use the case number provided. Case numbers are 8 digits in length, and you must include any leading zeros. For example, `00001234`.

Use the UI

Depending on the issue you're debugging, you can create a bundle either from the Leader or from an individual Edge Node.

UI Options

The **Create & Export Diag Bundle** modal provides these additional options:

- **Rename .js files**: When enabled, appends `.txt` to JavaScript files in the bundle. Defaults to toggled on.
- **Include metrics**: When enabled, includes the persistent internal metrics store used by Leaders and single-instance deployments in the bundle. This metrics store can grow to many gigabytes over time. Individual Edge Node do not maintain this metrics store, their metric-related information still appears in logs, diag artifacts, and other files regardless of this setting. Defaults to toggled on.
- **Include git**: Includes the last 14 days of git log in the bundle, identifying each commit's message, author, timestamp, and files changed. Defaults to toggled on. Unavailable on Leader-managed Stream Workers and Edge Nodes, as well as on Outpost Nodes.
- **Include top talkers**: Adds detailed metrics for the top 10 Sources, Destinations, Pipelines, Routes, and Packs (based on the last 24 hours) to the diagnostic bundle. These JSON files (such as, `top_sources.json`) contain in/out events and bytes, aiding in troubleshooting and performance analysis. Disabled by default to minimize bundle size, this requires Leader Node access to the `/system/metrics/query` API.
- **Latest number of jobs**: The number of jobs (and all tasks) the bundle will include. Defaults to empty, which means unlimited jobs.
- **Include installation logs**: Includes Windows MSI logs, only available on Windows deployments of Cribl Edge. Default is toggled off.
    - This option is not available for Linux deployments.
    - If MSI logs are available, the diag bundle will contain them. If no MSI logs are present, the bundle will not contain them even with this option toggled on.

Cribl Support will advise you on changing the defaults here. Because the Leader metrics store can be large, they often recommend disabling **Include metrics** unless they specifically request metrics store contents for your case.

From the Leader Node

To create a bundle from the Leader Node, go to **Settings** > **Global** > **Diagnostics** and select **Create Diagnostic Bundle**.

- To share the bundle with Cribl Support, toggle **Send to Cribl Support** on. Enter your case number, and then select **Export**.
- To download the bundle locally to your machine (on-prem only), select **Export**.

From Edge Nodes

To create a bundle from individual Edge Nodes, you must be able to access the Edge Node's UI. You can easily use the Leader to get authenticated access to each Edge Node, if you first enable the parent Fleet's **Teleport** setting:

1. In the Cribl Edge sidebar, select **Fleets**.
2. Make sure the relevant Fleet's **Teleport** toggle is on.
3. Select the displayed number of Edge Nodes to access the **Edge Nodes** page.
4. In the **GUID** column, select the link for the Edge Node you're debugging.
5. Look for the purple border to confirm that you've teleported to that Edge Node.
6. Select **Edge Node Settings** (top right) > **Diagnostics** (lower left) > **Diagnostic Bundle**, and select **Create Diagnostic Bundle**.
7. Use the appropriate **Export** options listed above for the Leader.

On on-prem deployments, new and previously created bundles are stored in `$CRIBL_HOME/diag`. They're also listed in the UI, where you can re-download them or share them with Cribl Support.

Use the CLI

To create a bundle using the CLI, use the `diag` command. (This option is on-prem only.) Depending on the issue you're debugging, you can create a bundle either from the Leader host or from an individual Edge Node host.

diag command CLI

```
# $CRIBL_HOME/bin/cribl diag
Usage: [sub-command] [options] [args]

Commands:
create       - Creates diagnostic bundle for this instance, args:
  [-d]                  - Run create in debug mode.
  [-j]                  - Do not append '.txt' to js files.
  [-k]                  - Include metrics for the top 10 Sources, Destinations, Pipel
  [-t <maxIncludeJobs>] - Maximum number of jobs (and all associated tasks) to includ
  [-M]                  - Exclude the persistent internal metrics store from the bund
  [-g]                  - Exclude git log from bundle.
  [-i]                  - Include install logs.
heapsnapshot - Generate heap snapshot of a Cribl process, args:
  [-p <pid>] - The pid of the process to dump the heap snapshot
list         - List existing diagnostic bundles
send         - Send diagnostics bundle to Cribl Support, args:
   -c <caseNumber> - Cribl Support Case Number
  [-p <path>]      - Diagnostic bundle path (if empty then new bundle will be created


## Creating a diagnostic bundle
# $CRIBL_HOME/bin/cribl diag create
Created Cribl Stream/Edge diagnostic bundle at /opt/cribl/diag/ <product>-<hostname>-

## Creating and sending a diagnostic bundle
# $CRIBL_HOME/bin/cribl diag send -c <caseNumber>
Sent Cribl diagnostic bundle to Cribl Support

## Sending a previously created diagnostic bundle
# $CRIBL_HOME/bin/cribl diag send -p /opt/cribl/diag/ <product>-<hostname>-<datetime>
Sent Cribl diagnostic bundle to Cribl Support
```

Copy

For complete command syntax, see diag.



Taking a diag from a Cribl Edge Node running on Windows requires setting the `CRIBL_VOLUME_DIR` environment variable. See `diag` for more information.

Include CPU Profiles

This section is applicable only to Cribl Stream.

If Cribl Support asks you to grab CPU profiles of Worker Processes, follow these steps:

1. Use `top` or `htop` on the Edge Node to identify Edge Node PIDs consuming a lot of CPU.
2. See Measuring CPU or API Load for instructions on accessing the UI's **Profile** options (for your deployment type), and generating and saving profiles.
3. Find the Worker Processes matching the PIDs you identified above.
4. Select **Profile** on each. Start with the default 10-second **Duration**.
5. Once the profile is displayed, save it to a `.json` file. (See details at the above link.)
6. Repeat steps 3-5 for other CPU-intensive Worker Processes.
7. Upload the profile `.json` files to Cribl Support.

On an already CPU-starved Worker Node, profiling might fail with an error message, or just hang. In this case, you might need a few retries to get a successful profile.

Include Memory Snapshots

Cribl.Cloud does not support taking memory snapshots. This feature is for on-prem deployments only.

If you encounter a memory leak, Cribl Support might request a memory snapshot of a `cribl` process to better understand where the leak is occurring.

There are two ways to generate a memory snapshot:

- Automatically
- Manually

Automatically Capture a Memory Snapshot

To automatically capture memory snapshots when Worker Processes approach or exceed memory limits, turn on the **Enable heap snapshots setting** in the Fleet's **General Settings**. Navigate to **Fleet Settings** > **System** > **Worker Processes** and toggle on. Only the two most recent heap snapshots are retained. Older snapshots are automatically deleted. This behavior cannot be modified.

This setting should only be enabled if recommended by support or if you are experiencing out-of-memory issues. Be aware that enabling this setting may impact system performance and storage usage.

For details, see Worker Processes.

Manually Capture a Memory Snapshot Prerequisites

To manually generate a reliable memory snapshot for Cribl Edge diagnostics, ensure two key conditions are met:

1. **Correct `CRIBL_HOME` Environment Variable**: This variable points to the Cribl Edge installation directory. Open a new terminal session and verify its value using `echo $CRIBL_HOME`. If empty, set it appropriately for your operating system (for example, `export CRIBL_HOME=/path/to/cribl` on Linux, `setex cribl C:\Program Files\Cribl /m` on Windows).
2. **Run as the `cribl` User**: The `diag heapsnapshot` command requires permissions to create the snapshot file within the run Cribl Edge directory. Use `sudo su - cribl` to switch to the cribl user before running the command. Remember to exercise caution with `sudo`. On Windows, if Cribl is set to run under a specific user account, log in as that user before running Cribl. Alternatively, you can run Cribl Edge as an `Administrator`.

Capture the Memory Snapshot

To capture the memory snapshot:

1. Note the PID of the process of interest.
2. In a terminal, `cd` to the directory where Cribl Edge is installed.

```
cd $CRIBL_HOME/bin
```

Copy

3. Run the `diag heapsnapshot` command with the PID you noted in step 1.

- For Linux:

```
./cribl diag heapsnapshot -p <pid>
```

Copy

Locate the PID on Windows

There are three methods to identify the Process ID (PID) for `cribl.exe`:

1. **Task Manager**: Open the Task Manager application. Locate the **Processes** tab and search for `cribl.exe` in the list. The PID will be displayed in the corresponding column.
2. **Command Line Interface (CLI)**: Open a Command Prompt window. Run the following command:

```
tasklist /FI "IMAGENAME eq cribl.exe"
```

Copy

This command will list all running processes. The line containing `cribl.exe` will also display the PID in the corresponding column (usually the second column).

3. **PowerShell**: Open a PowerShell window. Run the following command:

```
PowerShell
Get-Process -Name cribl
```

Copy

This command will list details about running processes. The output will include the PID for any process named `cribl`.

Once you locate the PID, run the `diag heapsnapshot` command:

1. Navigate to the Cribl Edge installation directory: For example, `C:\Program Files\Cribl\bin` or `$CRIBL_HOME\bin`.
2. Run the `diag heapsnapshot` command with the PID.

```
./cribl diag heapsnapshot -p <pid>
```

Copy

Cribl will capture the memory snapshot in `$CRIBL_HOME/diag/`.

Example output:

```
Heap-1690906031148-2458958.heapsnapshot
```