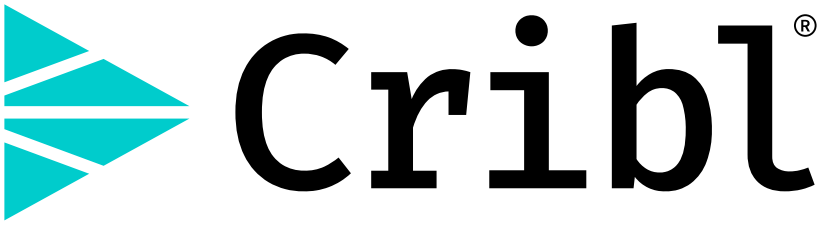


Cribl Edge Documentation Manual

Version: v4.9

Generated: 2024-12-04 00:59:32

Copyright (c) 2017- 2024 Cribl, Inc. All rights reserved.

1. Introduction 12
1.1. About Cribl Edge 12
1.2. Basic Concepts 14
1.3. Cribl.Cloud vs. Self-Hosted 18
1.4. Getting Started with Cribl Edge 23
1.5. Getting Started Tutorial 27
1.6. Explore Cribl Edge on Linux 34
1.7. Explore Cribl Edge on Windows 54
1.8. QuickConnect 70
2. Cribl.Cloud 76
2.1. Initial Cribl.Cloud Setup 78
2.2. Managing Cribl.Cloud 80
2.3. Cribl.Cloud Billing 82
2.4. Managing Cribl.Cloud Members 86
2.5. Connect On-Premises Leaders to Cribl.Cloud Billing 90
2.6. Enterprise Cloud 95
3. Deploying Edge 99
3.1. Deployment Planning 99
3.1.1. Planning Large-Scale Deployments of Cribl Edge 106
3.2. Installing Cribl Edge on Linux 109
3.2.1. Enabling Start on Boot 114
3.2.2. Running Edge as an Unprivileged User 118
3.2.3. Installing Edge on Linux via RPM 120
3.2.4. Anti-Virus Exceptions 126
3.3. Installing Cribl Edge on Windows 127
3.3.1. System Proxy Configuration on Windows 138
3.4. Running in a Docker Container 142
3.5. Deploying via Kubernetes 145
3.6. Leader High Availability/Failover 154
3.7. Ports 161
4. Fleet Management 164
4.1. Fleets 167
4.1.1. Explore Cribl Edge Fleets 167
4.1.2. Fleets Hierarchy and Design 172
4.1.3. Creating and Managing Fleets and Subfleets 176
4.1.4. Configuring Fleets and Deploying Changes 181

4.1.5. Recovering a Deleted Fleet 185
4.2. Edge Nodes 186
4.2.1. How Edge Nodes and Leader Work Together 186
4.2.2. Manage Edge Nodes 188
4.2.3. Add and Update Edge Nodes 191
4.2.4. Filter Edge Nodes 195
4.2.5. Mapping Edge Nodes to Fleets 202
4.2.6. Managing Edge Node Config Bundles 206
4.2.7. Set Up Leader and Edge Nodes 208
5. Cribl Edge Better Practices 214
5.1. Configuring Cribl Edge 214
5.2. Fleets Hierarchy and Design 223
5.3. Monitoring your Infrastructure with Cribl Edge 227
5.4. Kubernetes Observability Using Cribl Edge 232
5.5. Windows Observability Using Cribl Edge 241
5.6. Monitor and Troubleshoot Cribl Edge 254
6. Workspaces 258
6.1. Configuring Workspaces 262
7. Upgrading Edge 264
7.1. Administering 264
7.1.1. Upgrading Overview 264
7.1.2. Better Practices for Upgrading Cribl Edge 265
7.1.3. Upgrading via the UI 268
7.1.3.1. Upgrading Edge Leader via UI 268
7.1.3.2. Upgrading Edge Nodes via UI 272
7.1.3.3. Upgrading Edge on Windows via the UI 274
7.1.3.4. Monitoring Node Upgrade Status 275
7.1.3.5. Fleet Upgrading Use Case 277
7.1.3.6. Backup & Rollback via the UI 278
7.1.4. Upgrading Manually 279
7.1.4.1. Upgrading Manually Overview 279
7.1.4.2. Upgrading Cribl Edge on Linux (CLI) 281
7.1.4.3. Upgrading RPM Installations of Cribl Edge 284
7.1.4.4. Manually Upgrading Windows Edge Nodes 285
7.1.4.5. Manual Rollback 287
7.1.5. Upgrading Containers 289
7.1.6. Troubleshooting Edge Upgrades 292

7.2. Notifications Overview 294
7.2.1. Source-State Notifications 298
7.2.2. Destination-State Notifications 302
7.2.3. License-Expiration Notifications 307
7.2.4. Email Notifications 309
7.2.5. Notification Targets 313
7.2.5.1. Webhook Notification Targets 315
7.2.5.2. PagerDuty Notification Targets 322
7.2.5.3. Slack Notification Targets 324
7.2.5.4. AWS SNS Notification Targets 326
7.2.5.5. Email Notification Targets 330
7.3. Licensing 333
7.4. Uninstalling Cribl Edge from Linux 340
7.5. Uninstalling Cribl Edge from Windows 341
7.6. Fleet Settings 342
7.7. Customizing the interface 349
7.7.1. Custom Banners 349
7.7.2. Custom Login Page 350
7.8. Scripts 351
8. Access Management 353
8.1. Authentication 355
8.2. Members and Permissions 362
8.3. Teams 368
8.4. Local Users 369
8.5. Roles 371
8.6. Using ACLs to Allow Cribl Edge to Read Files 380
8.7. Cribl.Cloud SSO 383
8.7.1. SSO 383
8.7.1.1. Cribl.Cloud SSO Setup 383
8.7.1.2. Configuring SSO groups 384
8.7.1.3. SSO with Okta and OIDC (Cribl.Cloud) 387
8.7.1.4. SSO with Microsoft Entra ID and SAML (Cribl.Cloud) 391
8.7.1.5. SSO with Okta and SAML (Cribl.Cloud) 396
8.7.1.6. SSO Troubleshooting in Cribl.Cloud 400
8.7.2. On-Prem SSO 401
8.7.2.1. SSO with Okta and OIDC (On-Prem) 401
8.7.2.2. SSO with Okta and SAML (On-Prem) 407

8.7.2.3. SSO with Microsoft Entra ID and SAML (On-Prem) 412
8.7.2.4. SSO with Microsoft Entra ID and OIDC (On-Prem) 416
9. Securing 420
9.1. Secure Your Cribl.Cloud Deployment 420
9.2. Secure Your On-Prem/Hybrid Deployment 423
9.3. Secure Leader/Edge Nodes Communication 429
9.3.1. How to Secure the Auth Token for the Leader Node 434
9.3.2. TLS Defaults and System-wide Settings 437
9.3.3. Securing Cribl.Cloud with TLS and Mutual TLS 440
9.3.4. Configure mTLS Authentication On-Prem 442
9.3.5. Secure Leader Using an Existing TLS/SSL Certificate and Key 445
9.3.6. Configure TLS for API and UI Access 447
9.3.7. Secure Cribl Edge Sources and Destinations with Certificates 449
9.3.8. Configure System Proxy 451
9.3.9. Configure SOCKS Proxy 456
9.4. Manage Secrets and Keys 458
9.4.1. Create and Manage Encryption Keys 458
9.4.2. Create and Manage Secrets in Cribl Edge 460
9.4.3. Configure KMS 462
10. Integrations 468
10.1. Sources 469
10.1.1. System 475
10.1.1.1. Exec Source 475
10.1.1.2. File Monitor Source 480
10.1.1.3. Journal Files Source 489
10.1.1.4. Kubernetes Events Source 493
10.1.1.5. Kubernetes Logs Source 498
10.1.1.6. Kubernetes Metrics Source 505
10.1.1.7. System Metrics Source 510
10.1.1.8. System State Source 518
10.1.2. Internal 529
10.1.2.1. Cribl HTTP Source 529
10.1.2.2. Cribl TCP Source 537
10.1.2.3. Cribl Stream Source (Deprecated) 543
10.1.2.4. Cribl Internal Source 549
10.1.2.5. Datagen Source 554
10.1.3. Amazon 556

10.1.3.1. Amazon Data Firehose Source 556
10.1.4. Prometheus 564
10.1.4.1. Prometheus Edge Scraper Source 564
10.1.4.2. Prometheus Remote Write Source 572
10.1.4.3. Grafana Source 580
10.1.4.4. Loki Source 588
10.1.5. Splunk 596
10.1.5.1. Splunk HEC Source 596
10.1.5.2. Splunk TCP Source 611
10.1.6. Windows 620
10.1.6.1. Windows Event Forwarder Source 620
10.1.6.1.1. Client Certificate Authentication for Windows Event Forwarder 636
10.1.6.1.2. Kerberos Authentication for Windows Event Forwarder 639
10.1.6.2. Windows Event Logs Source 644
10.1.6.3. Windows Metrics Source 649
10.1.7. AppScope Source 655
10.1.8. Datadog Agent Source 663
10.1.9. Elasticsearch API Source 674
10.1.10. HTTP/S (Bulk API) Source 685
10.1.11. Raw HTTP/S Source 696
10.1.12. Metrics Source 704
10.1.13. Model Driven Telemetry Source 711
10.1.14. NetFlow Source 716
10.1.15. OpenTelemetry (OTel) Source 729
10.1.16. SNMP Trap Source 738
10.1.17. Syslog Source 745
10.1.18. TCP JSON Source 756
10.1.19. TCP (Raw) Source 763
10.1.20. UDP (Raw) Source 770
10.2. Destinations 776
10.2.1. Managing Destinations 782
10.2.2. Internal 786
10.2.2.1. Cribl HTTP Destination 786
10.2.2.2. Cribl TCP Destination 795
10.2.2.3. Disk Spool Destination 803
10.2.2.4. Cribl Stream Destination (Deprecated) 806
10.2.2.5. Default Destination 811

10.2.2.6. DevNull Destination 812
10.2.2.7. Output Router Destination 813
10.2.3. Amazon 816
10.2.3.1. Amazon CloudWatch Logs Destination 816
10.2.3.2. Amazon Kinesis Data Streams Destination 823
10.2.3.3. Amazon S3 Compatible Stores Destination 831
10.2.3.4. Amazon SQS Destination 841
10.2.3.5. Amazon Security Lake Destination 848
10.2.4. Azure 855
10.2.4.1. Azure Blob Storage Destination 855
10.2.4.2. Azure Data Explorer Destination 862
10.2.4.3. Azure Event Hubs Destination 874
10.2.4.4. Azure Monitor Logs Destination 883
10.2.4.5. Microsoft Sentinel Destination 890
10.2.5. Elastic 900
10.2.5.1. Elasticsearch Destination 900
10.2.5.2. Elastic Cloud Destination 909
10.2.6. Google Cloud 916
10.2.6.1. Google Security Operations (SecOps) Destination 916
10.2.6.2. Google Cloud Logging Destination 925
10.2.6.3. Google Cloud Pub/Sub Destination 932
10.2.6.4. Google Cloud Storage Destination 938
10.2.7. Kafka 945
10.2.7.1. Kafka Destination 945
10.2.7.2. Confluent Cloud Destination 956
10.2.7.3. Amazon MSK Destination 967
10.2.8. Metrics 978
10.2.8.1. Graphite Destination 978
10.2.8.2. StatsD Destination 983
10.2.8.3. StatsD Extended Destination 988
10.2.9. New Relic Ingest 993
10.2.9.1. New Relic Events Destination 993
10.2.9.2. New Relic Logs & Metrics Destination 1000
10.2.10. Prometheus 1008
10.2.10.1. Prometheus Destination 1008
10.2.10.2. Grafana Cloud Destination 1015
10.2.10.3. Loki Destination 1024

10.2.11. Splunk 1031
10.2.11.1. Splunk HEC Destination 1031
10.2.11.2. Splunk Single Instance Destination 1040
10.2.11.3. Splunk Load Balanced Destination 1047
10.2.12. CrowdStrike Falcon LogScale Destination 1059
10.2.13. ClickHouse Destination 1066
10.2.14. CrowdStrike Falcon Next-Gen SIEM Destination 1074
10.2.15. Datadog Destination 1080
10.2.16. Dynatrace HTTP Destination 1088
10.2.17. Dynatrace OTLP Destination 1096
10.2.18. Exabeam Security Operations Platform Destination 1103
10.2.19. Filesystem/NFS Destination 1109
10.2.20. Honeycomb Destination 1115
10.2.21. InfluxDB Destination 1121
10.2.22. MinIO Destination 1128
10.2.23. NetFlow Destination 1137
10.2.24. OpenTelemetry (OTel) Destination 1140
10.2.25. SentinelOne DataSet Destination 1149
10.2.26. ServiceNow Cloud Observability 1157
10.2.27. SignalFx Destination 1166
10.2.28. SNMP Trap Destination 1172
10.2.29. Sumo Logic Destination 1175
10.2.30. Syslog Destination 1183
10.2.31. TCP JSON Destination 1196
10.2.32. Wavefront Destination 1203
10.2.33. Webhook Destination 1209
10.3. Destination Backpressure Triggers 1226
10.4. Persistent Queues 1234
10.4.1. Planning for Persistent Queues 1239
10.4.2. Configuring Persistent Queues 1246
10.5. About Load Balancing 1253
10.6. Cribl Edge to Cribl Stream 1255
10.7. Integrating with Other Services 1259
10.7.1. Cribl Edge as a Sidecar Container in AWS ECS 1259
10.7.2. Amazon Security Lake Integration 1265
10.7.3. Amazon Elastic Kubernetes Services (EKS) Add-On for Edge 1274
11. Monitoring Health and Metrics 1279
11.1. Internal Metrics 1295

11.2. Internal Logs 1304
11.3. Tracking Data Activity for an Edge Node 1310
12. Working with Data 1315
12.1. Event Model 1315
12.2. Event Processing Order 1317
12.3. Routes 1319
12.4. Pipelines 1325
12.4.1. Create Pipelines Using Cribl Copilot 1332
12.5. Packs 1335
12.5.1. Packs Publication Standards 1354
12.6. Use Datagens 1363
12.7. Data Preview 1368
12.8. Data Onboarding 1383
13. Functions 1387
13.1. Auto Timestamp 1392
13.2. Aggregations 1397
13.3. CEF Serializer 1406
13.4. Chain 1408
13.5. Clone 1410
13.6. Code 1412
13.7. Comment 1415
13.8. DNS Lookup 1416
13.9. Drop 1421
13.10. Dynamic Sampling 1422
13.11. Eval 1425
13.12. Event Breaker 1429
13.13. Flatten 1432
13.14. GeoIP 1435
13.15. Grok 1438
13.16. JSON Unroll 1440
13.17. Lookup 1442
13.18. Mask 1448
13.19. Numerify 1454
13.20. OTLP Logs 1456
13.21. OTLP Metrics 1466
13.22. OTLP Traces 1491
13.23. Parser 1498
13.24. Publish Metrics 1509

13.25. Redis 1515
13.26. Regex Extract 1527
13.27. Regex Filter 1531
13.28. Rename 1532
13.29. Rollup Metrics 1535
13.30. Sampling 1537
13.31. Serialize 1539
13.32. SNMP Trap Serialize 1542
13.33. Suppress 1545
13.34. Tee 1548
13.35. Unroll 1551
13.36. XML Unroll 1553
13.37. Prometheus Publisher (deprecated) 1557
13.38. Reverse DNS (deprecated) 1559
13.39. Trim Timestamp (deprecated) 1561
14. Knowledge Libraries 1563
14.1. Lookups Library 1564
14.2. Event Breakers 1568
14.3. Parsers Library 1582
14.4. Global Variables Library 1584
14.5. Regex Library 1586
14.6. Grok Patterns Library 1589
14.7. Schemas Library 1590
14.8. Parquet Schemas 1593
14.9. AppScope Configs 1602
15. Reference 1610
15.1. API Docs 1610
15.2. HTTP Response Codes 1619
15.3. CLI Reference 1620
15.4. Config Files 1644
15.4.1. cribl.yml 1648
15.4.2. breakers.yml 1654
15.4.3. certificates.yml 1655
15.4.4. groups.yml 1656
15.4.5. inputs.yml 1657
15.4.6. instance.yml 1732
15.4.7. jobs.yml 1733
15.4.8. job-limits.yml 1752

15.4.9. leader.yml 1753
15.4.10. licenses.yml 1755
15.4.11. limits.yml 1756
15.4.12. logger.yml 1758
15.4.13. mappings.yml 1759
15.4.14. messages.yml 1760
15.4.15. outputs.yml 1761
15.4.16. parsers.yml 1822
15.4.17. perms.yml 1823
15.4.18. policies.yml 1824
15.4.19. regexes.yml 1827
15.4.20. roles.yml 1828
15.4.21. samples.yml 1829
15.4.22. schemas.yml 1830
15.4.23. scripts.yml 1831
15.4.24. secrets.yml 1832
15.4.25. service.yml 1833
15.4.26. vars.yml 1834
15.5. Cribl Expressions 1835
15.5.1. C.Crypto – Data Encryption and Decryption 1836
15.5.2. C.Decode and C.Encode - Encoding and Decoding 1839
15.5.3. C.Lookup – Inline Lookup Methods 1844
15.5.4. C.Mask – Data Masking Methods 1848
15.5.5. C.Net – Network Methods 1855
15.5.6. C.Text – Text Methods 1861
15.5.7. C.Time – Time Methods 1865
15.5.8. Miscellaneous Expression Methods 1871
15.6. Cribl Expression Syntax 1878
15.7. Environment Variables 1881
15.8. Linux System Metrics Details 1888
15.9. Windows System Metrics Details 1905
16. Troubleshooting 1943
16.1. Known Issues 1943
16.2. Get Product Help 2037
16.3. Common Challenges on the Edge 2038
16.4. Diagnose Issues 2046

1. Introduction

1.1. About Cribl Edge

All the docs to goat you started with Cribl Edge.

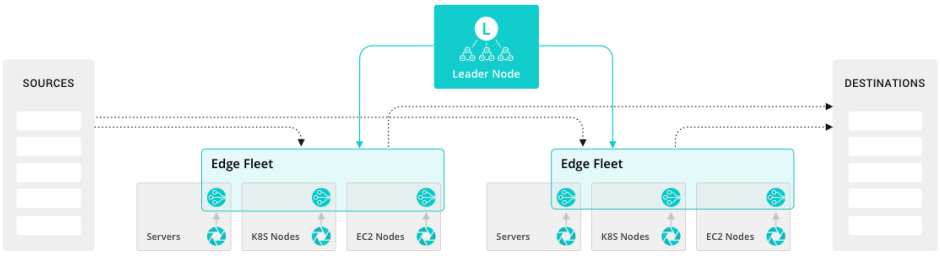
Download all docs as a PDF – 4.9.2 | Download Binary | Edge on Cribl.Cloud

- Questions not answered here? We'd love to help you. Meet us in #Cribl Community Slack – sign up here.

What Is Cribl Edge?

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

- Automatic discovery of host, container and application metrics, logs, etc. on endpoints.
- Great Getting Data In (GDI) experience: explore, preview, build configs etc.
- Collect, Process and Forward
    - Collect logs, metrics (host, process, container), traces (coming soon), custom command output etc.
    - All the processing power of functions & pipelines at your fingertips
    - Forward data to any of the many supported destinations
- Centralized Configuration & Management
    - Visual configuration authoring, version controlled etc.
    - Teleportation-to-the-edge for local preview and configuration validation



Edge Leader, Fleets, and Destinations

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.

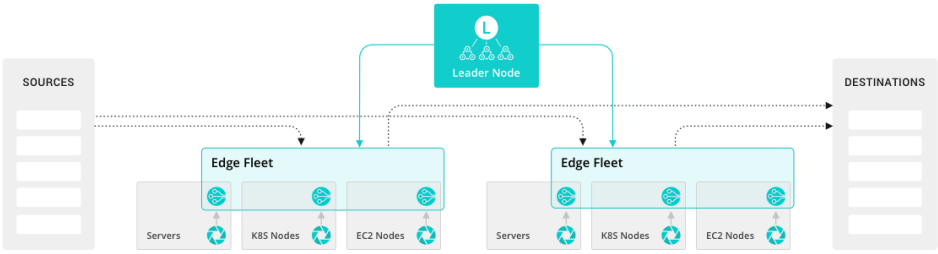
Cribl University offers the Cribl Certified Admin for Edge course where you will learn to implement, optimize, and manage your deployments. To follow the direct course link, first log into your Cribl University account. (To create an account, select the **Sign up** link. You'll need to click through 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.

1.2. Basic Concepts

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

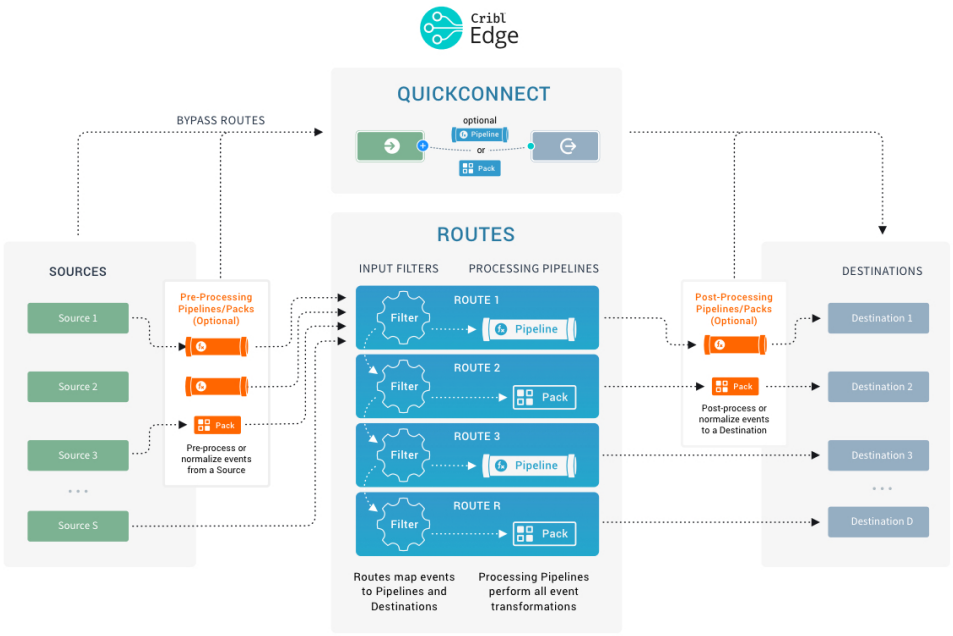
---

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 and Windows 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

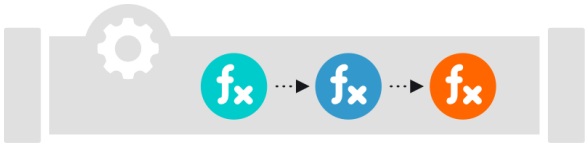
Routes evaluate incoming events against **filter** expressions to find the appropriate Pipeline to send them to. Routes are **evaluated in order**. Each Route can be associated **with only one** Pipeline and one output (configured as a Edge **Destination**).

By default, each Route is created with its `Final` flag set to `Yes`. With this setting, a Route-Pipeline-Destination set will consume events that match its filter, and that's that.

However, if you disable the Route's `Final` flag, 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.

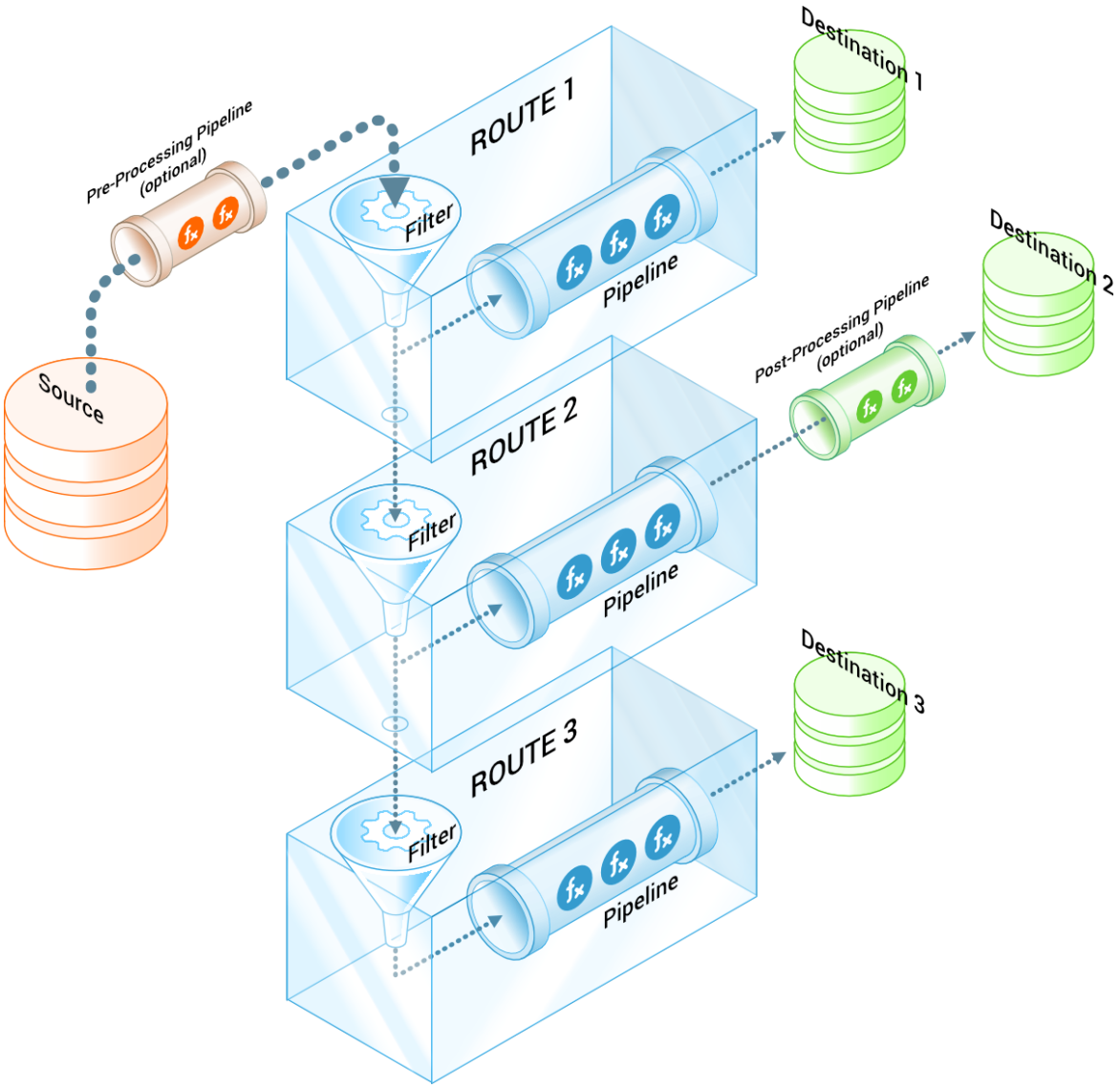
Pipelines

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.

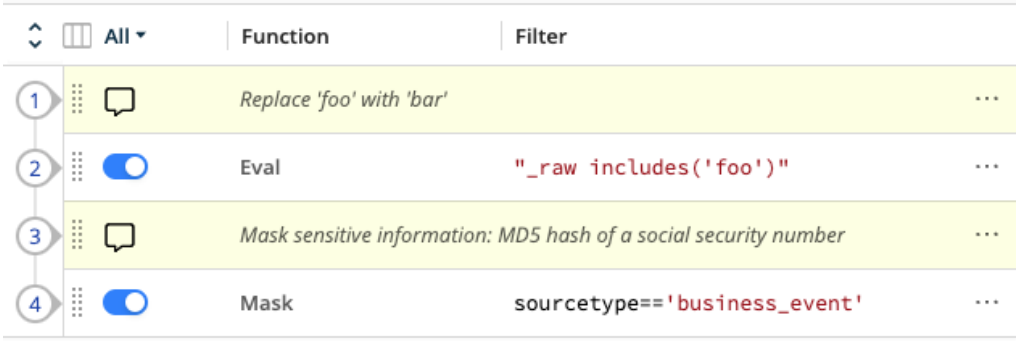


Pipelines categorized by position

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. For more details, see Functions.

1.3. Cribl.Cloud Vs. Self-Hosted

A Cribl.Cloud deployment differs in several ways from a customer-managed (on-prem) deployment of Cribl Edge 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.

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 for Cribl Stream or Cribl Edge. A Free or Standard plan allows only a single Fleet, and some Distributed Settings cannot be configured.

With an Enterprise plan, Cribl always provides at least two Workers, and will scale up further Workers as needed to meet your peak load. With an Enterprise plan, you also have the option to configure additional

hybrid Edge Nodes and Fleets.

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 portal. 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 Member 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.

Feature	Self-Hosted	Cribl.Cloud
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.	Pre-configured 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. This is because your license is managed by your parent Cribl.Cloud portal, where you can check credits and usage history on the Billing tab.

Other Simplified Settings

These features are available only on customer-managed hybrid Cribl Stream Workers:

- Script Collector
- Staging directory support in File-based Destinations
- Tee Function
- File System Collector
- Filesystem Destination

These features are available only in on-prem deployments:

- **Settings** > **Global** > **Scripts** (if enabled – Cribl.Cloud does not support configuring or running shell scripts).

These features are unavailable on Cribl-managed Stream Workers in Cribl.Cloud:

- System State Source
- AppScope Source's **Filter Settings**

Configuring persistent queues in a Cribl.Cloud deployment requires an Enterprise plan. On customer-managed (hybrid or on-prem) Worker Groups and on all Edge Nodes, you can freely define the **Max queue size**, based on the disk space you provision.

However, on Cribl-managed Worker Groups, each Source or Destination's queue is allocated a maximum of 1 GB disk space per Worker Process. (Given this automatic configuration, Cribl-managed Sources and Destinations expose only limited PQ controls.)

Available Ports and TLS Configurations

To get data into Cribl.Cloud, your Cribl.Cloud portal 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.

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 connected to the same Leader.

This traffic does not count against your ingestion quota, so this routing prevents double-billing. (For related details, see Exemptions from License Quotas.)

Simplified Source, Collector, and Destination Configuration

Several commonly used Sources are preconfigured for you within Cribl.Cloud's UI, and are ready to use.

The Exec Source is unavailable on Cribl-managed Workers in Cribl.Cloud, but is available on hybrid Workers.

The Cribl Internal Source's CriblLogs option, when used on Cribl-managed Worker Groups in Cribl.Cloud, contains only logs related to Sources and Destinations.

1.4. Getting Started With 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 install 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 continue, make sure your machine meets the minimum deployment requirements for Edge. Edge Nodes should have sufficient CPU, RAM, network, and storage capacity to handle your **specific** workload. It's very important to test this before deploying to production.

Requirement Type	Requirements Details
Minimum: Edge Nodes	**OS**: Linux: RedHat, CentOS, Ubuntu, AWS Linux, Suse, Windows Server (64bit) **System**: ~1Ghz processor, 512MB RAM, 5GB of free disk space (more if persistent queuing is enabled on Edge Nodes)
Recommended: Leader Node	**OS**: Linux: RedHat, CentOS, Ubuntu, AWS Linux, Suse (64bit) **System**: +4 physical cores, +8GB RAM, 5GB free disk space

Edge Nodes can't be installed on Windows laptops/desktops.

Download and Install Cribl Edge

To install Cribl Edge, select the Linux or Windows tab and follow the steps.

Linux Windows

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/
```

4. Un-tar the resulting `.tgz` file in the `/opt/` directory.

```
tar xvzf cribl-<version>-<build>-<arch>.tgz
```

   For example:

```
tar xvzf cribl-4.3.1-12f82b6a-linux-x64.tgz
```

   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
```

6. In your terminal, go to `/opt/cribl-edge/bin` and switch your installation to Edge mode:

```
./cribl mode-edge
```

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 Cribl Edge 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)`
    - `Asia Pacific (Sydney)`
    - `Canada (Central)`

For details on Cribl.Cloud Organizations and on managing users, refer to Cribl.Cloud documentation:

- Registering a Cribl.Cloud Portal.
- Managing Cribl.Cloud.

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.

1.5. Getting Started Tutorial

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 licence will be enough to go through all the steps.

Add the File Monitor Source

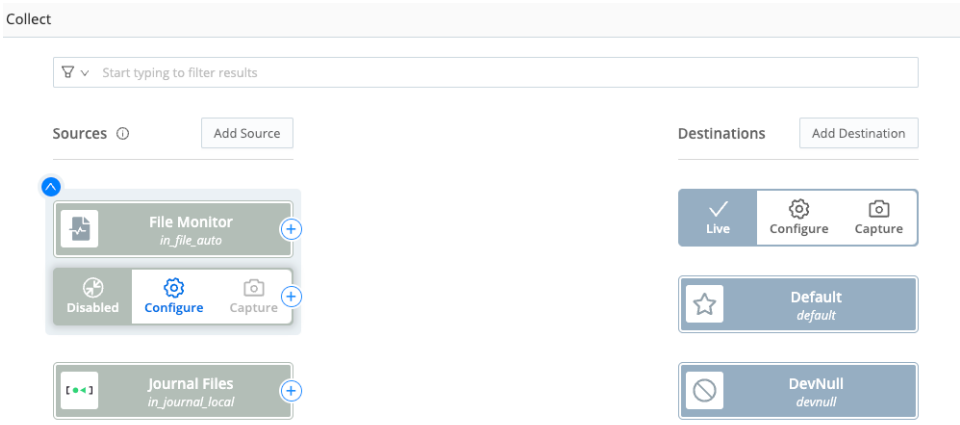
Each Cribl Edge Source represents a data input. Options include System Metrics, File Monitor, Kubernetes Metrics, Logs, and Events, as well as Prometheus Edge Scraper, and others.

For this tutorial, we'll use the File Monitor Source which generates events from text files or log files.

1. In Cribl Edge, click **Manage** > **Collect**. You can use the QuickConnect screen to configure how data flows between Sources and Destinations.

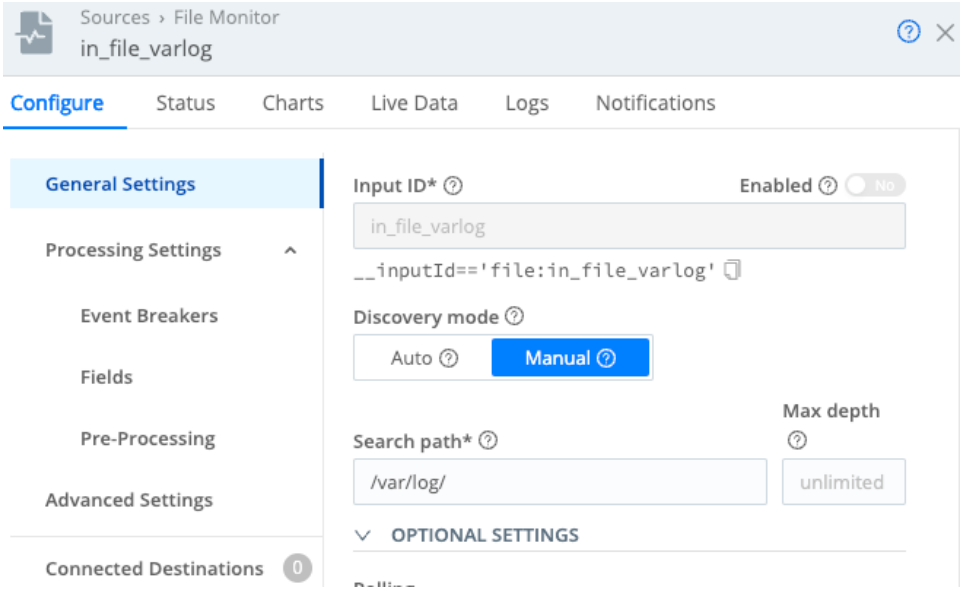
   If you're on Cribl.Cloud or any other Distributed mode, first click the `default_fleet` in its tile to select it.

2. From the list of Sources, expand the File Monitor tile and click **Configure** for **in_file_varlog**.



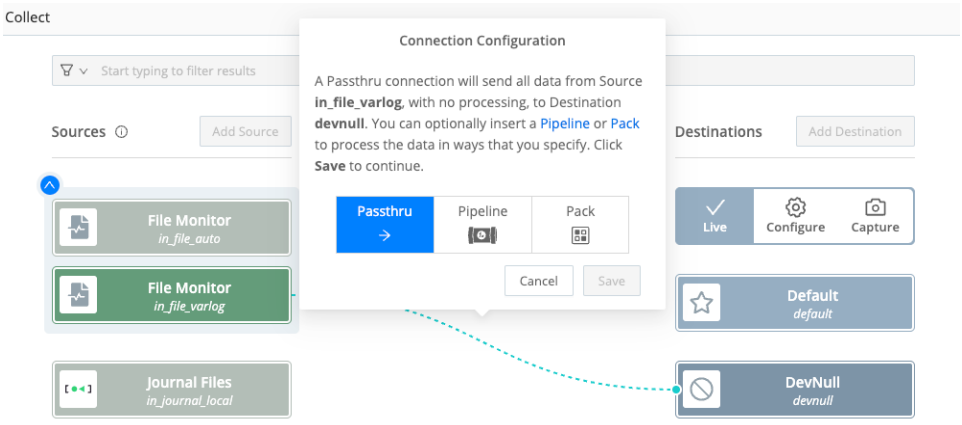
QuickConnect UI with the built-in File Monitor Sources.

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.

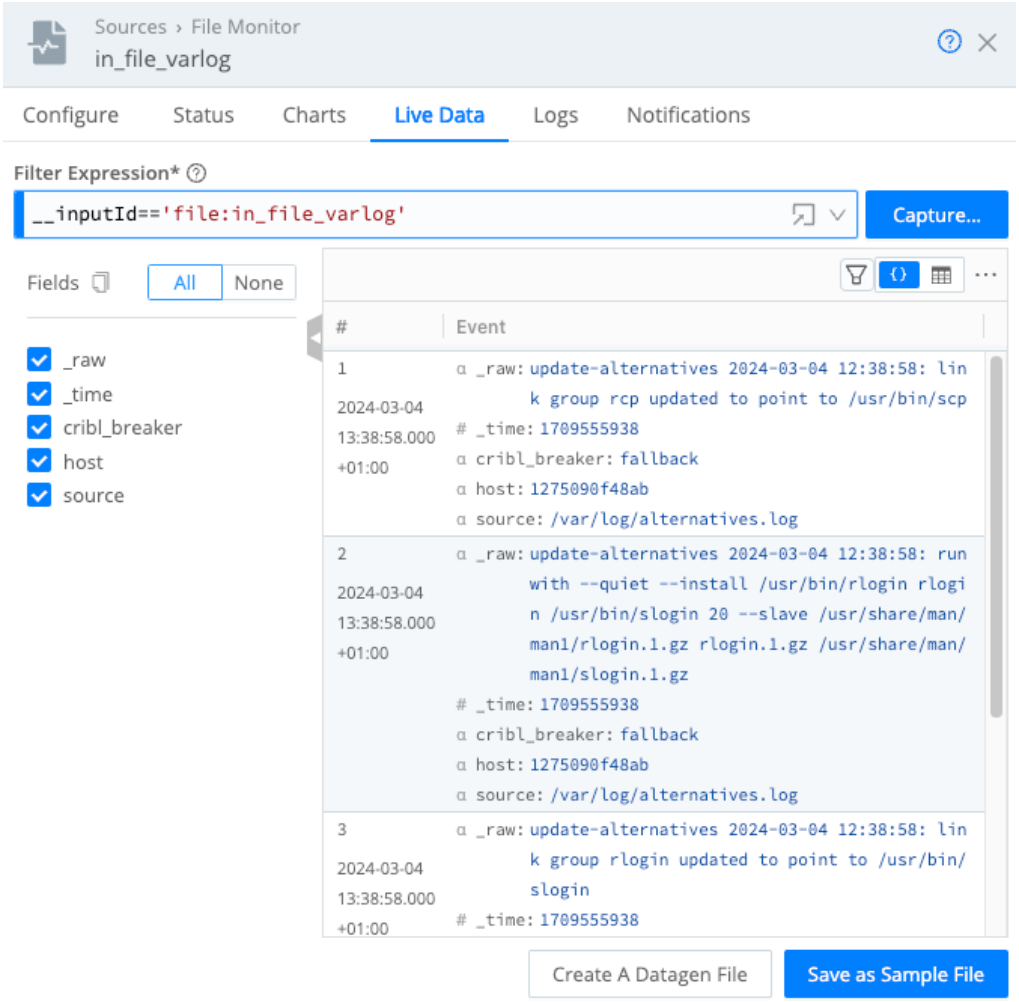


QuickConnect UI with a new connection between File Monitor and DevNull.

7. Click **Save**.

If you're on Cribl.Cloud or any other Distributed mode, click **Commit** at Cribl Edge's upper right before proceeding. Then, in the resulting dialog, type a commit message (for example, "Added Source") and click **Commit and Deploy** to confirm. You'll see a **Commit successful** message.

Now click **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 click **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 click **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 Splunk, Kafka, Kinesis, InfluxDB, Grafana Cloud, TCP JSON, and others.

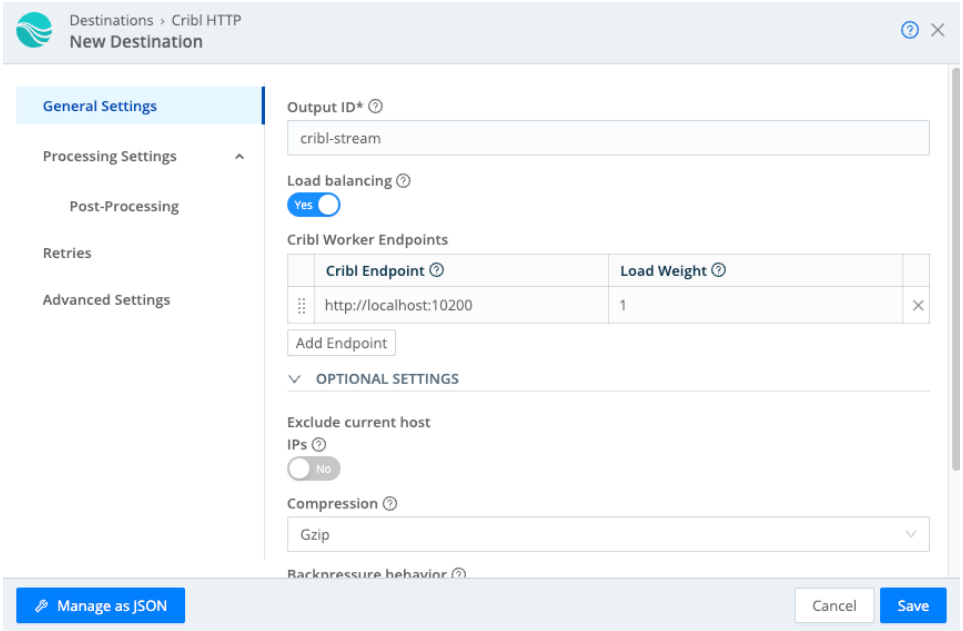
So far you've temporarily used the **DevNull** Destination. Now, let's set up an actual Destination where you'll sent the data to a separate Cribl Stream instance.

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 click the `default_fleet` in its tile to select it.

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. Click **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, click **Commit** at Cribl Edge's upper right before proceeding. Then, in the resulting dialog, type a commit message (for example, "Added Destination") and click **Commit and Deploy** to confirm. You'll see a **Commit successful** message.

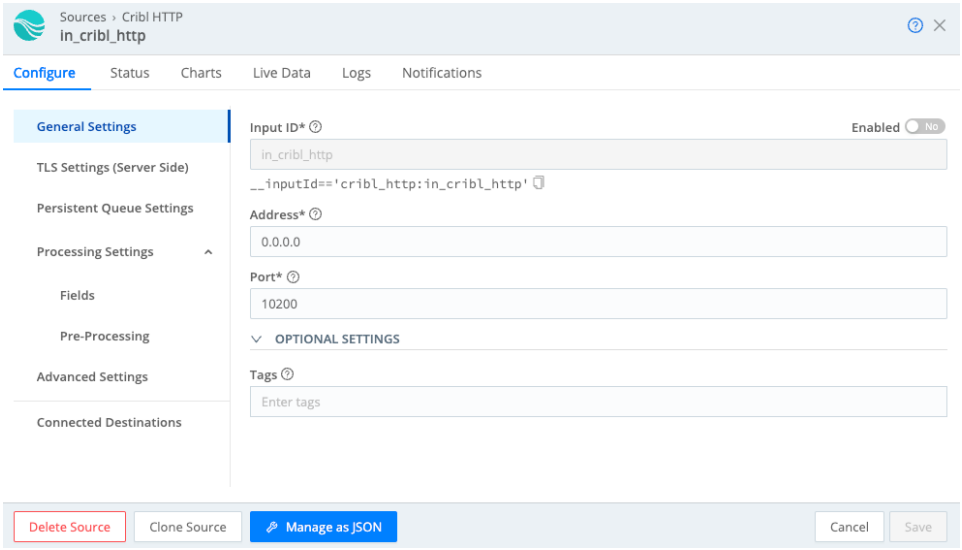
Receive Data

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 click the `default` (or any other) Worker Group.
2. In the top menu, select **Data** > **Sources**.
3. Search for **Cribl HTTP** and click 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.

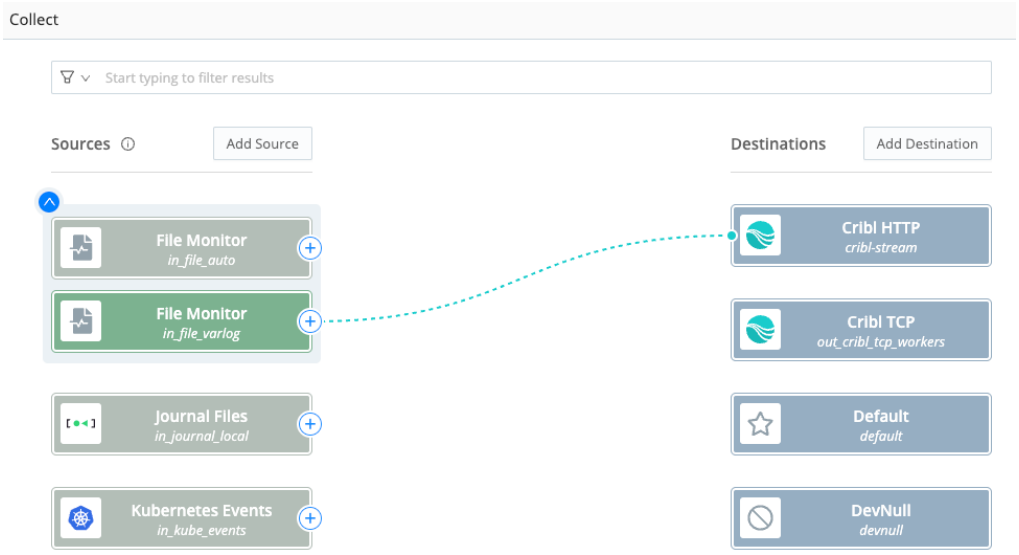
4. In the Source's configuration modal, make sure that the **Address** Field points to `0.0.0.0`, which is a catch-all for all addresses, and that **Port** is the same you configured for the Destination in Edge (using the example above, `10200`).



This Source will pick up events sent by Cribl Edge.

5. Enable the Source and exit the modal.
6. Click **Commit** at Cribl Stream's upper right. In the resulting dialog, type a commit message and click **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, click **Commit** at Cribl Edge's upper right before proceeding. Then, in the resulting dialog, type a commit message (for example, "Added Destination") and click **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 click 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** to `No`.
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 Sandboxes: 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.

That's it! Enjoy using Cribl Edge.

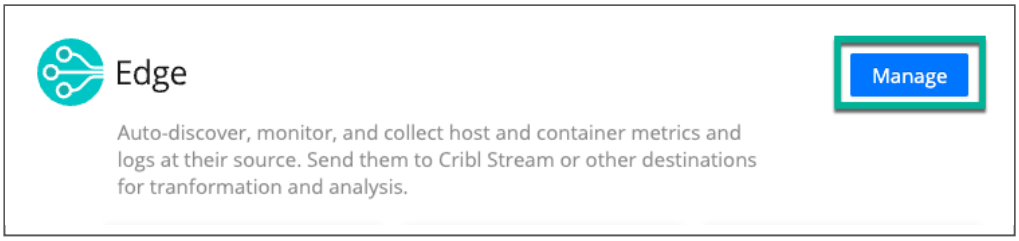
1.6. Explore Cribl Edge On Linux

The Cribl Edge UI offers a centralized view to manage, configure, and version–control your Edge Nodes. It also endows you with teleport–to–the–edge superpowers for locally previewing and validating your configurations. Here's a quick tour of the Cribl Edge UI in distributed mode.

Access Cribl Edge

When you first log into Cribl Edge (single-instance or distributed), you'll see tiles that prompt you to choose between the Cribl Stream or Cribl Edge UI. The Edge tile displays basic configuration details, including the number of Fleets, Subfleets, Edge Nodes, and events and bytes over time.

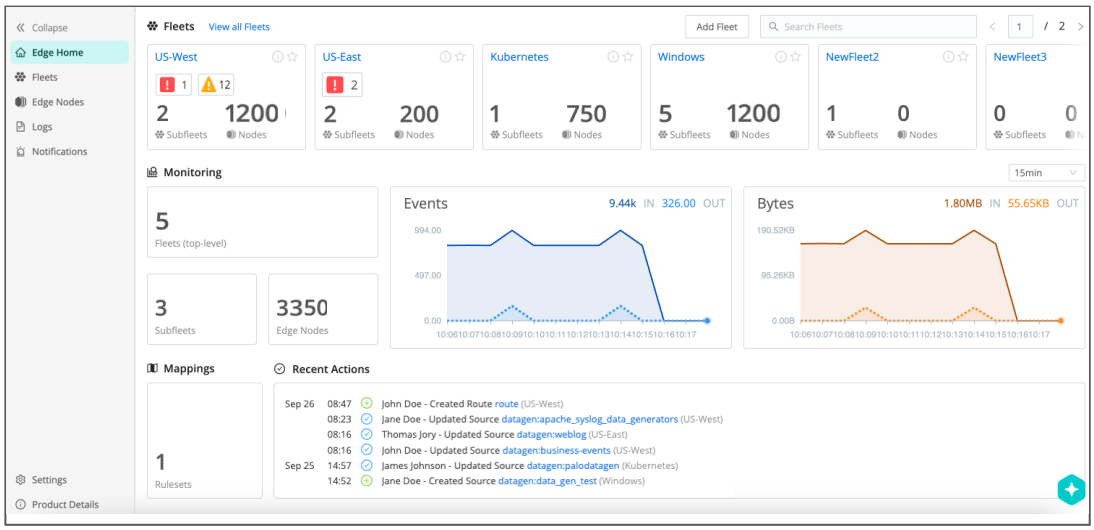
Select **Manage** to jump into Cribl Edge.



Manage Edge

Edge Home

Once you open Cribl Edge, you'll be on the **Edge Home** landing 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.



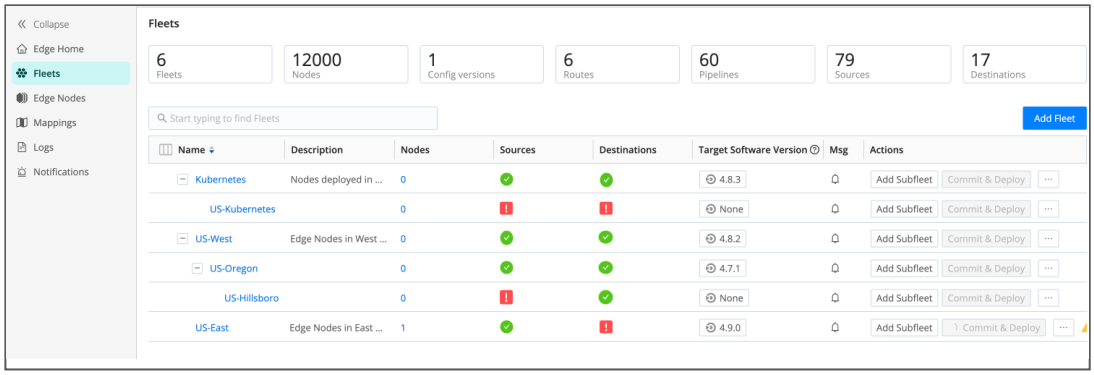
The Edge Home page, showing an overview of the configured resources

From the sidebar, you can quickly access more Cribl Edge goodies:

- Fleets and Subfleets
- Edge Nodes
- Mappings
- Logs
- Notifications

Fleets

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



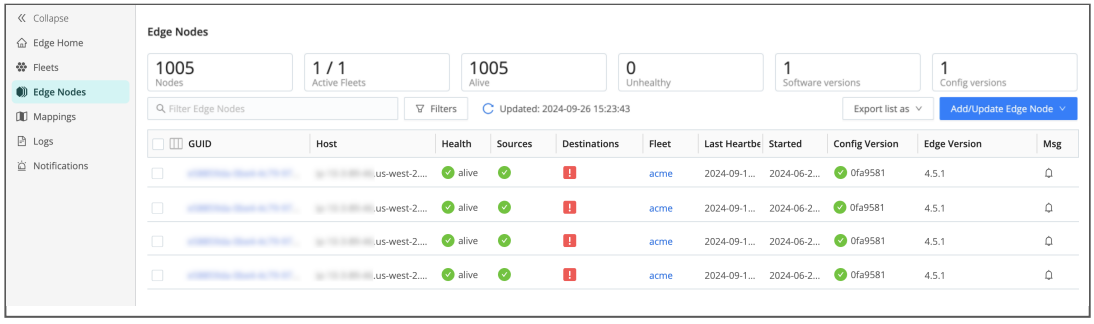
Fleets page

From here, you can select a Fleet **Name** to isolate individual Fleets, or use the **Search** bar to locate your Fleet.

- See Exploring Fleets for more information about the Fleets interface.
- For more information 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.



The list provides status information for each Edge Node in the selected Fleet, gives you the opportunity to filter Edge Nodes based on selected criteria, as well as a UI for adding and updating Nodes. For more information, see Manage Edge Nodes.

Mappings

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

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 our Managing Notifications docs.

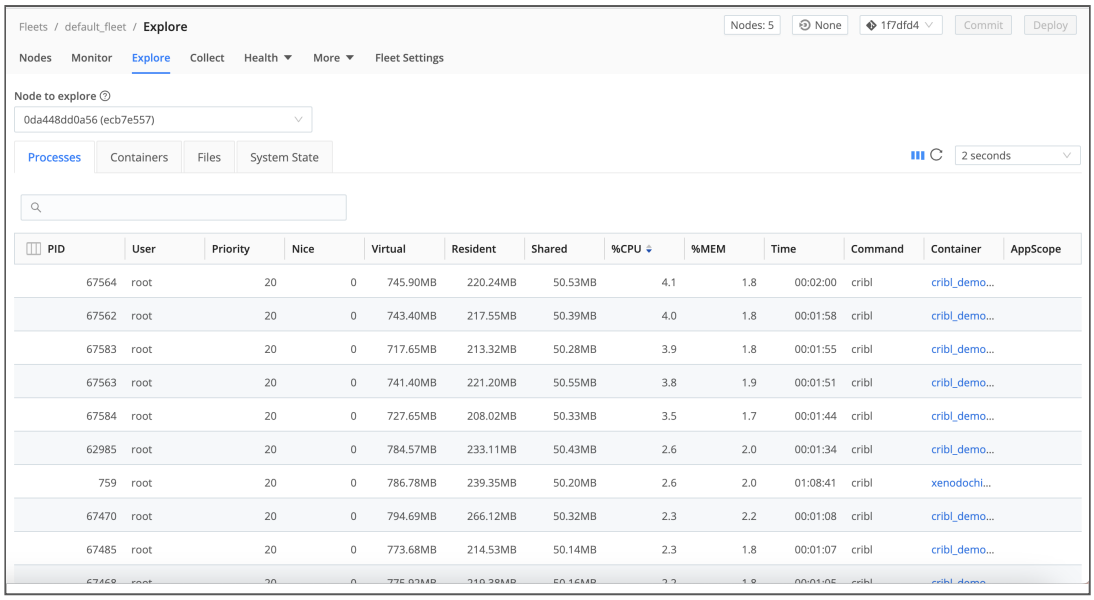
Logs

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.

Explore Tab

Select **Explore** to view more details on a particular Edge Node. On the **Node to explore** drop-down, select one of your hosts to display the following tabs:

- Processes
- Containers
- Files
- System State



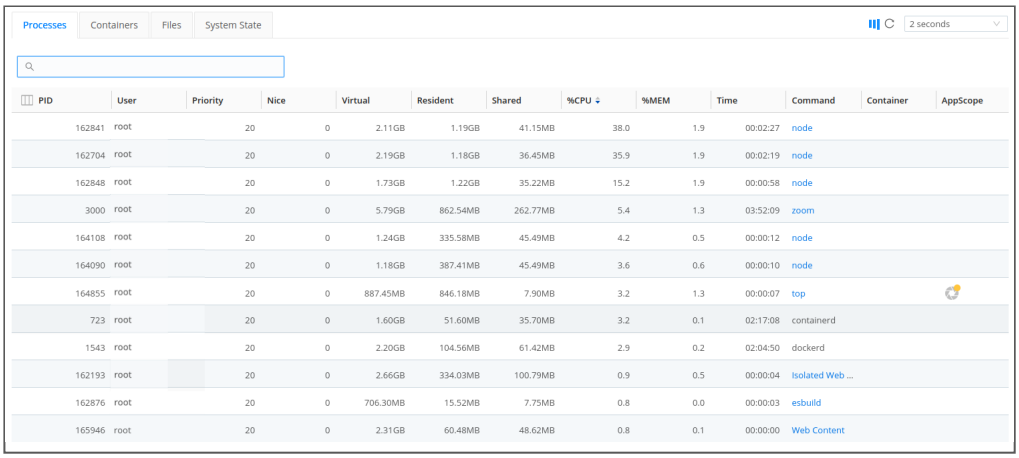
Explore an Edge Node

The **Node to explore** drop-down lists up to 50 Edge Nodes, ordered by hostname. To view the details for a specific Edge Node, enter the hostname or GUID into the **Node to explore** field.

Let's explore each tab.

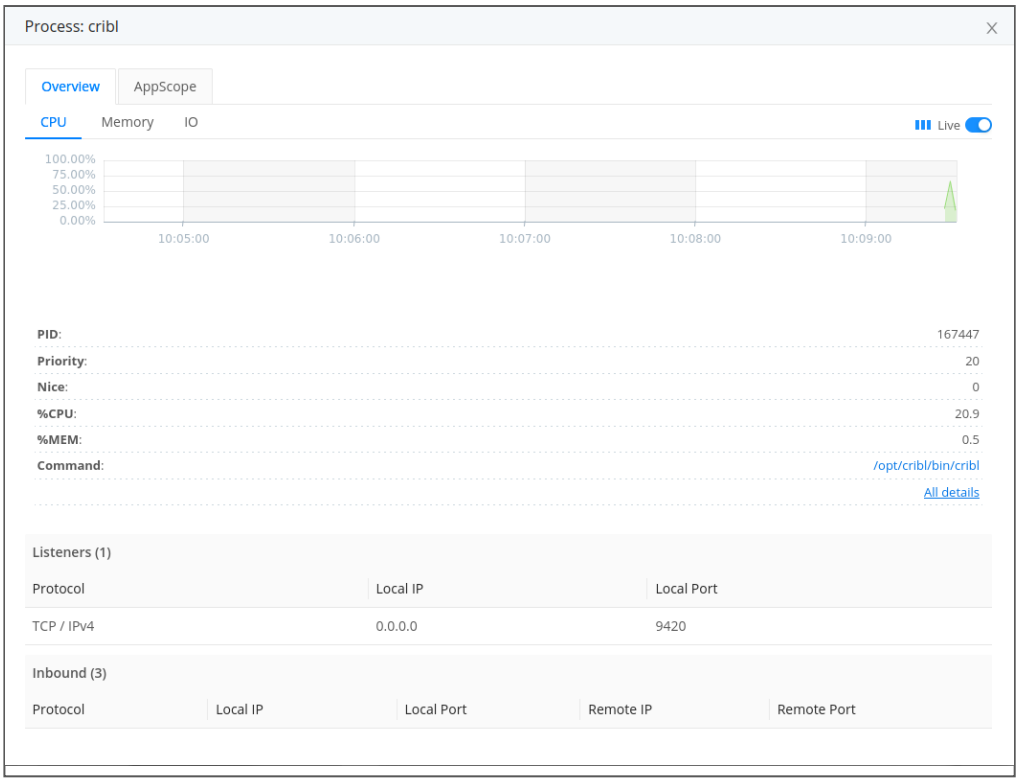
Processes

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



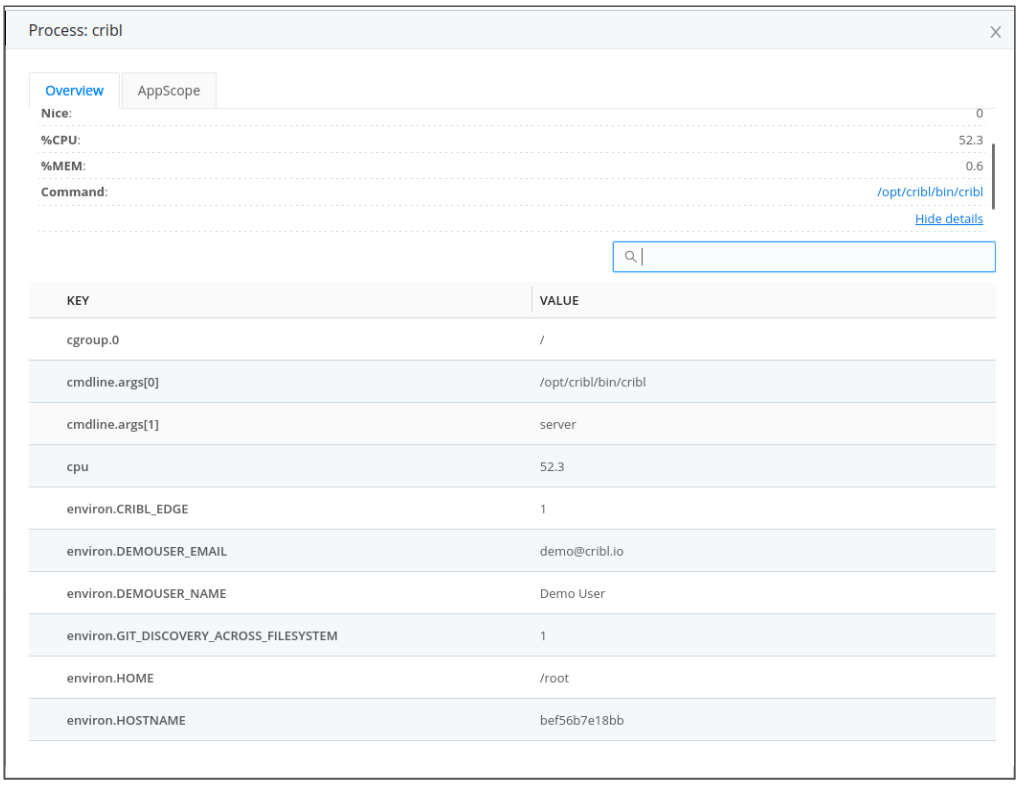
View Processes

Select on any of the rows to open the **Process: <process_name>** drawer. In the drawer's default **Overview** tab, you'll find basic information on the process, including CPU, Memory, and IO graphs, along with tables for active **Listening**, **Inbound**, and **Outbound** connections.



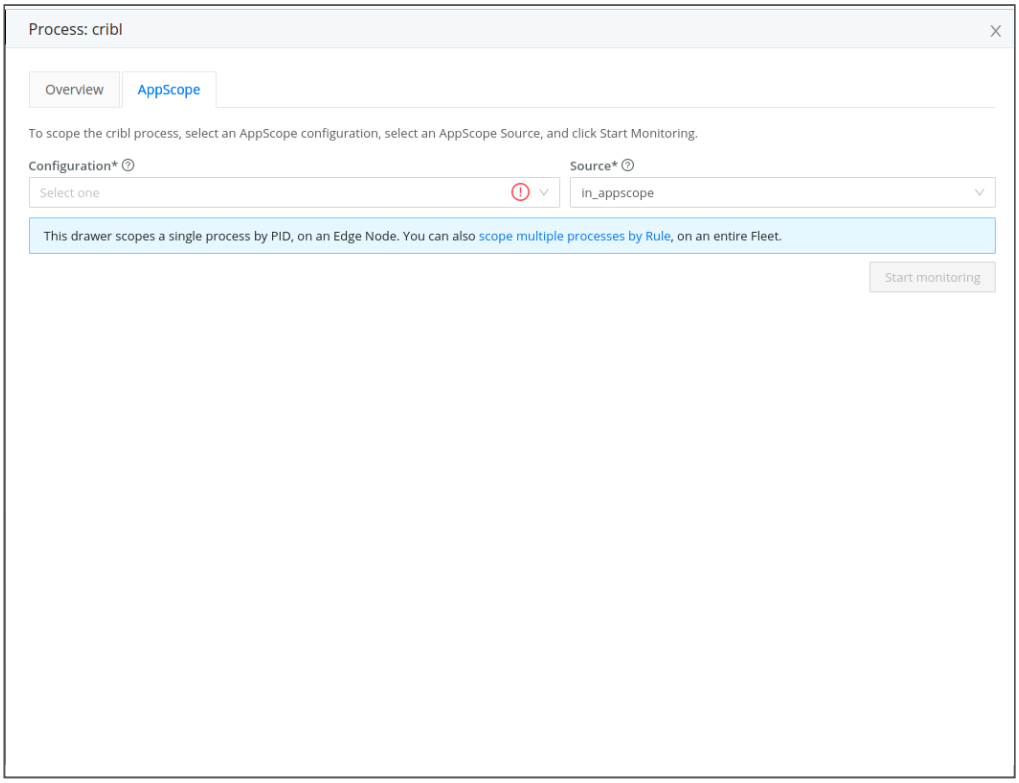
The Overview tab

In this tab, select **All details** to see the selected process' information out of `/proc`, expressed as key-value pairs. This information would normally require SSH'ing to the machine; this view makes troubleshooting across multiple systems much easier.



Showing all details of a process

Open the **AppScope** tab if you want to "scope" the process (i.e., use AppScope to monitor it). Once in the tab, you'll choose an an AppScope configuration that says what events and metrics to obtain, and an AppScope Source to receive them. See Scoping by PID in the AppScope docs. Note that your Cribl Edge instance must be running as root to do process monitoring with AppScope.



The AppScope tab for a process

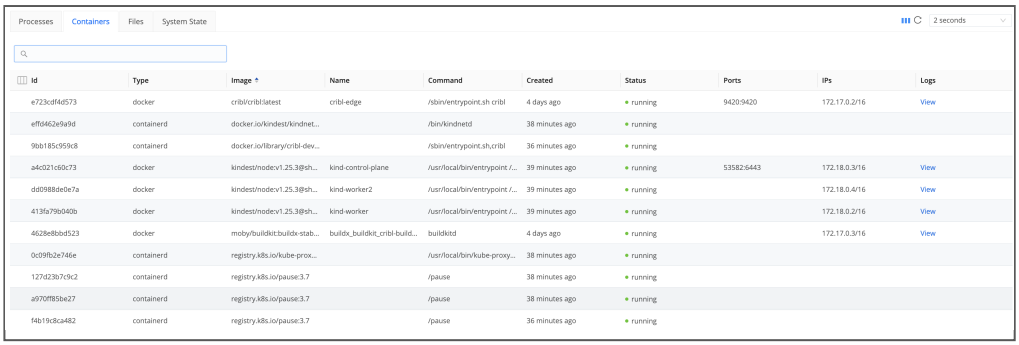
The **AppScope** column indicates when a process is being scoped, which you can view from the **Processes** tab.

Containers

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

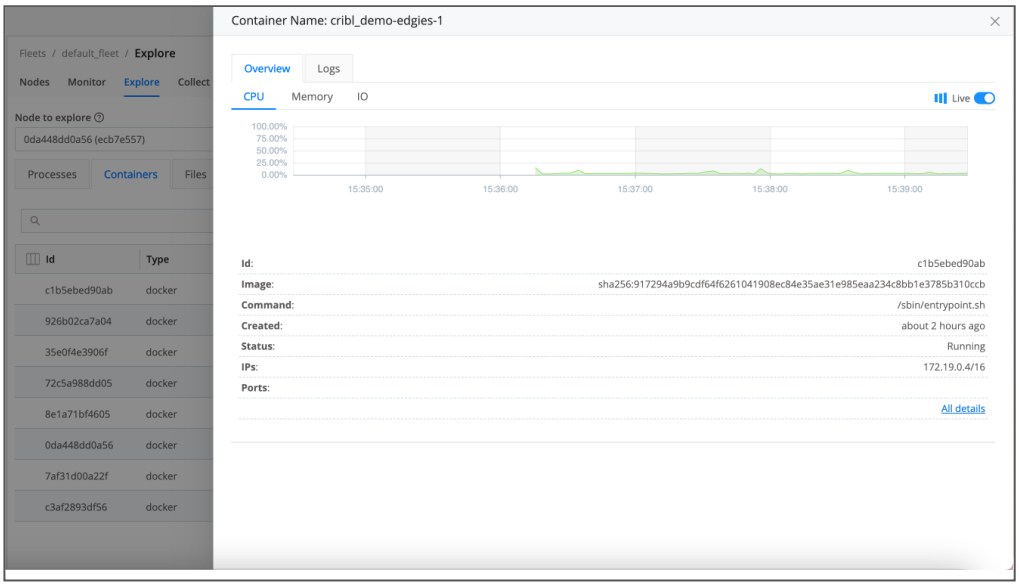
Cribl Edge supports both Docker and `containerd` runtimes.

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



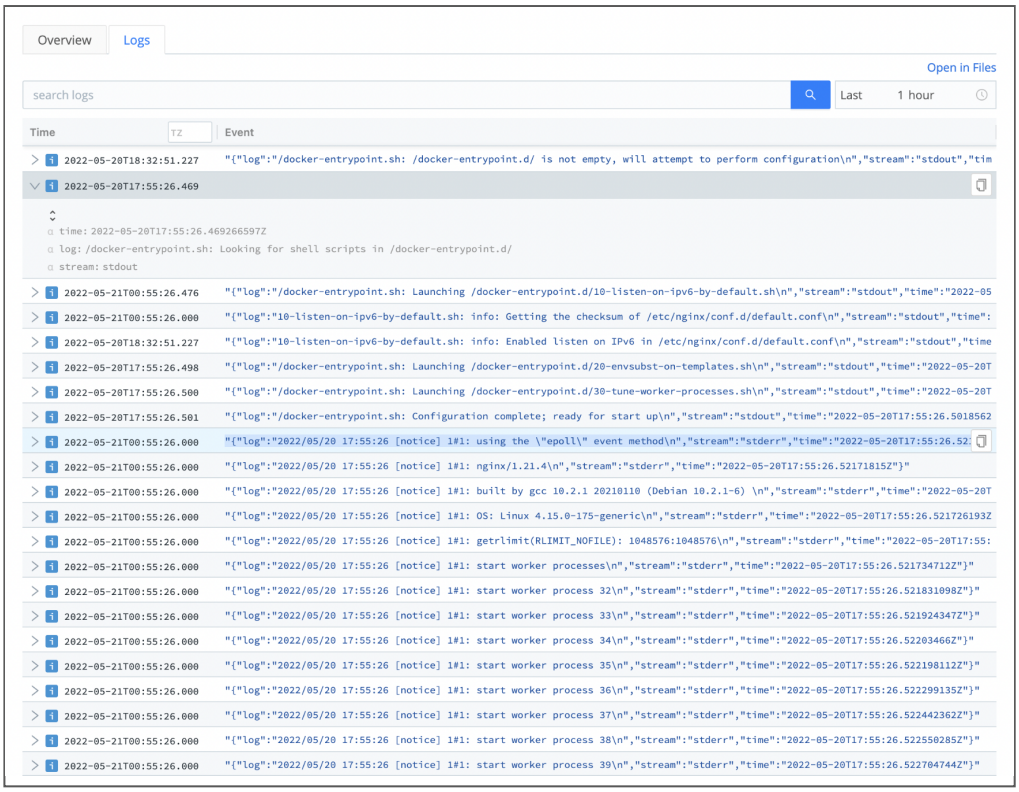
Containers overview

Select any container to view more details:



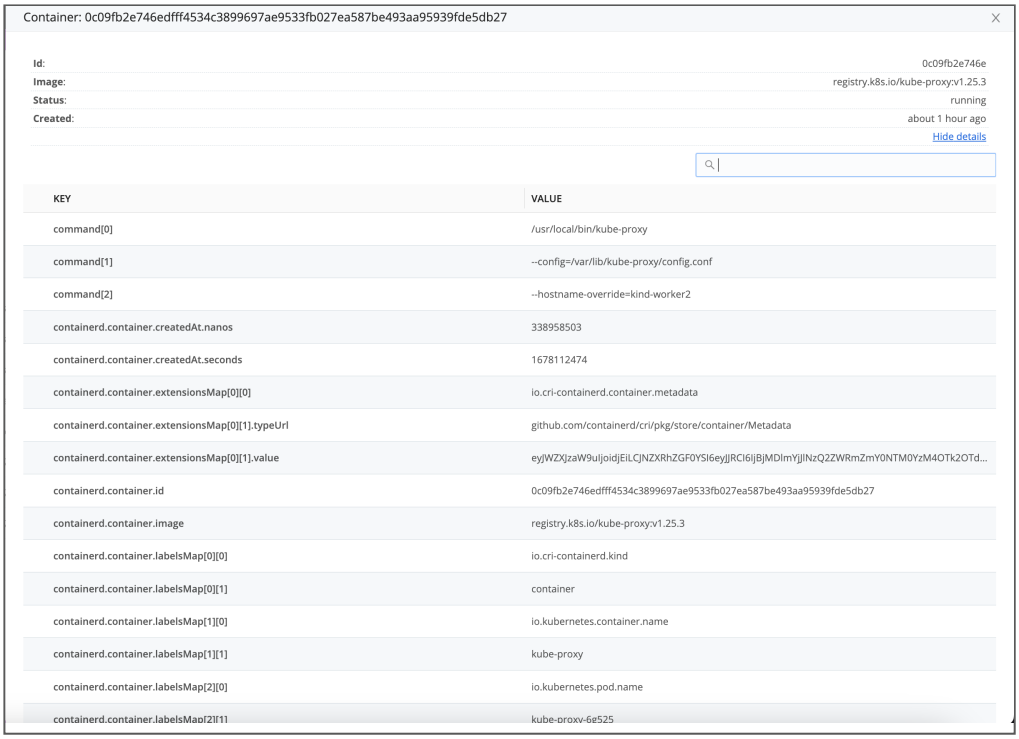
Container details

Select the **Logs** tab to view container logs. Optionally, use the search bar to filter displayed logs by arbitrary strings.



View Docker Container logs

The screenshot below shows `containerd` details, which don't include charts or logs.

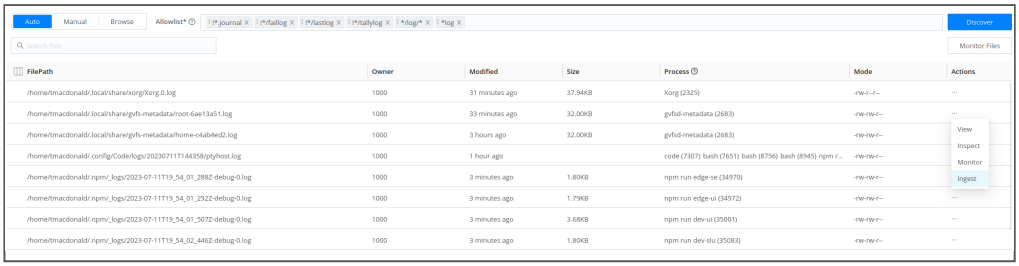


View containerd logs

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.



Explore files

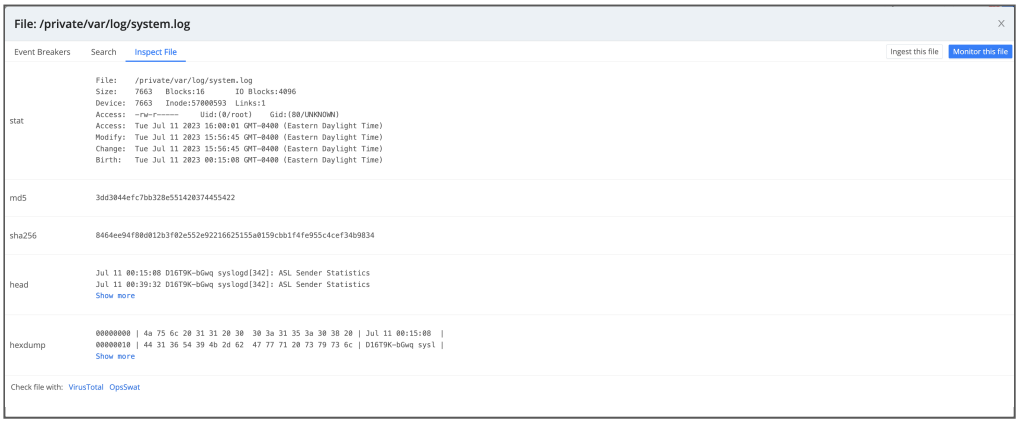
The **Actions** column allows you to:

- **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 if the file is flagged as compromised.



Inspect File tab

- **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

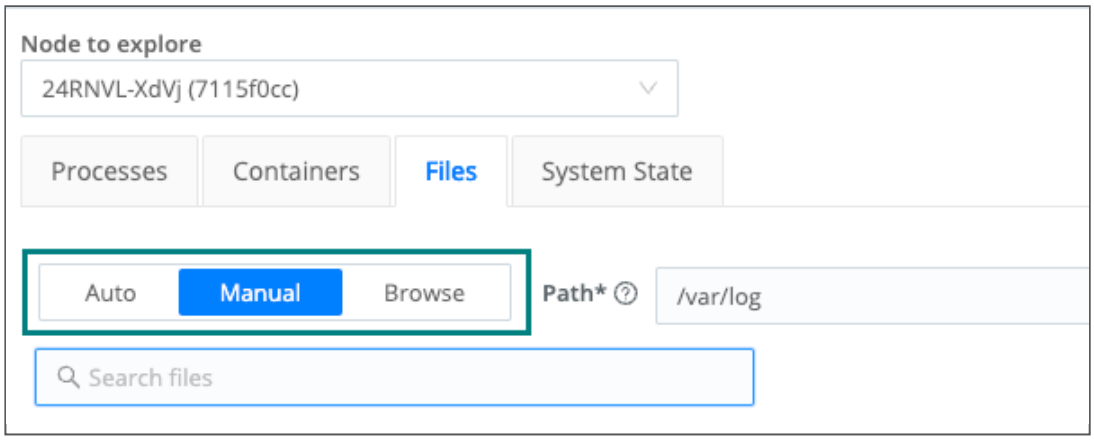
configurations.

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.
- **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.



Manual discovery mode

Path

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

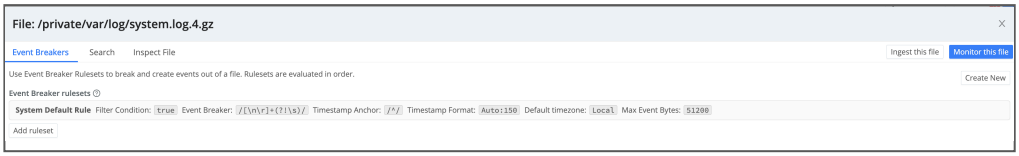
Allowlist

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.

Search tab

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



Event Breakers tab

Monitor Files

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

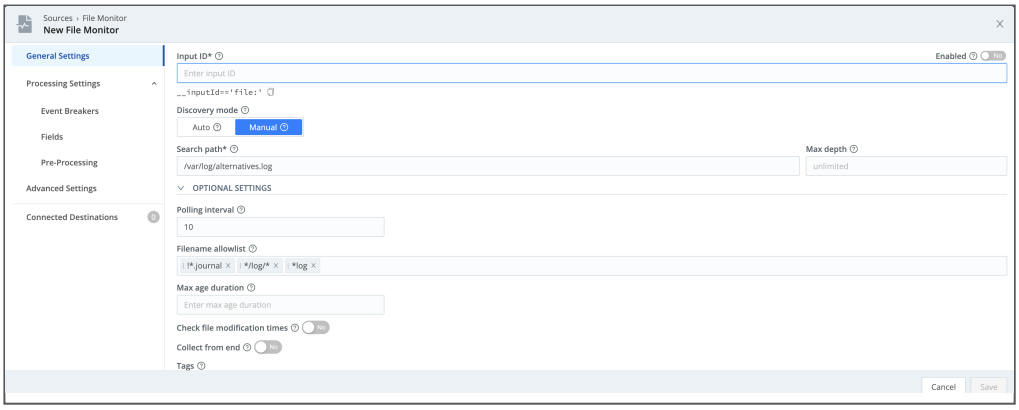
Max Depth

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.



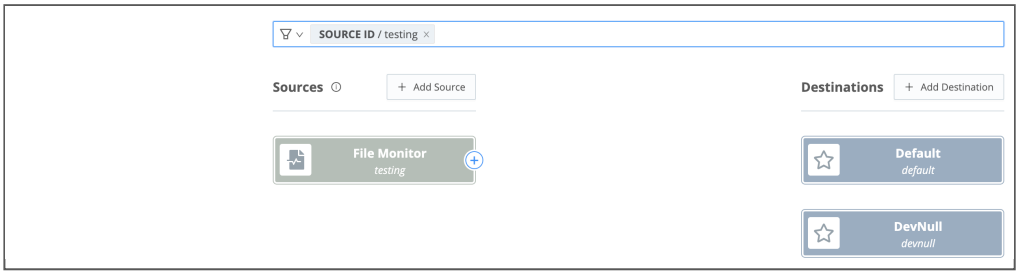
New File Monitor modal

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

- **Discovery mode**

- **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.



Collect page

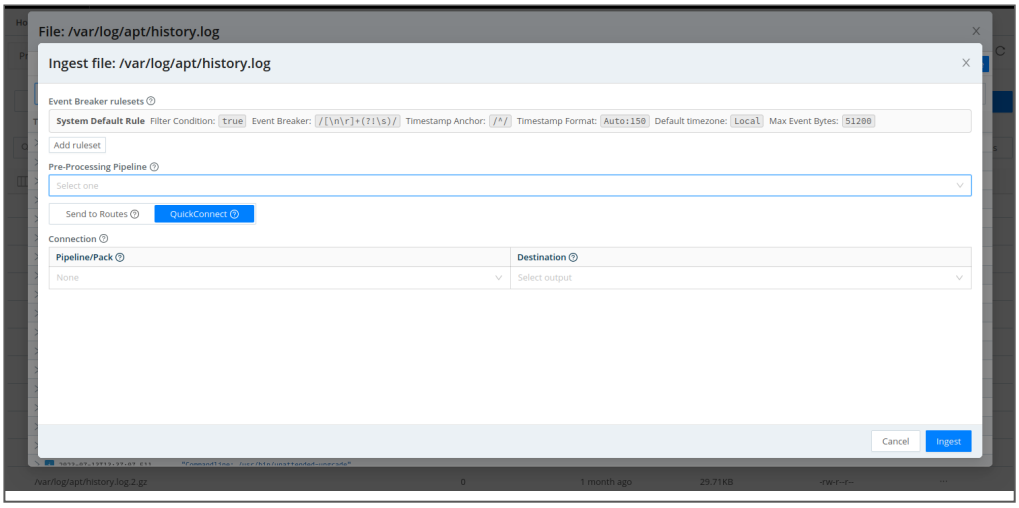
For further details, see File Monitor and QuickConnect.

Ingest a File

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

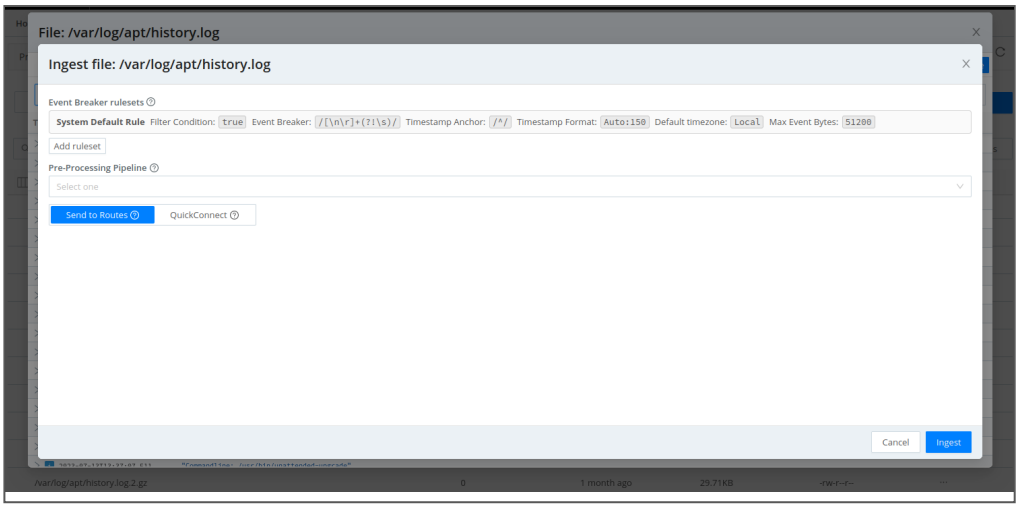
You have two options:

- Send directly to a configured Destination via **QuickConnect** (the default).



Ingest file via QuickConnect

- Send to Routes through (an optional) Pre-Processing Pipeline.



Ingest file via Routes

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 ruleset, select **Create New** to open the **New Ruleset** modal. Proceed as described here. Later, you can persist 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 these left tabs:

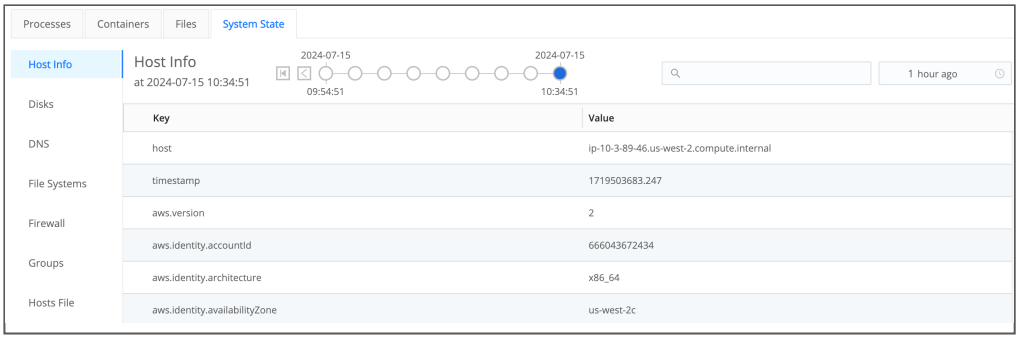
- Host Info
- Disks
- DNS
- File Systems
- Firewall

- Groups
- Hosts File
- Interfaces
- Listening Ports
- Logged-In Users
- Routes
- Services
- Users

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

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**.



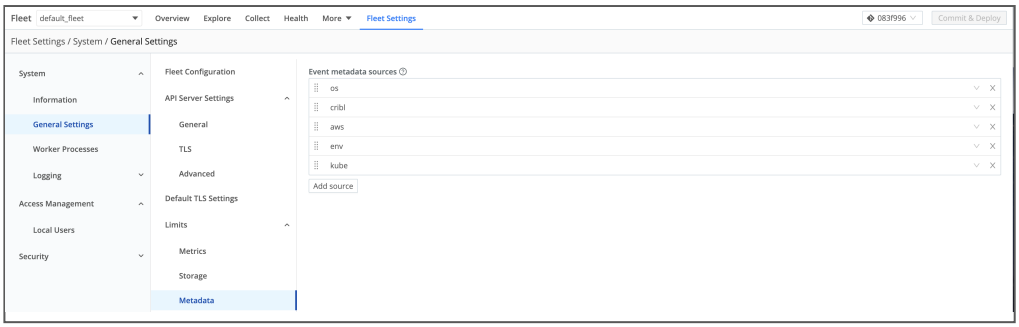
Host info/metadata collected

The metadata surfaced by an Edge Node can be used to:

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

You can customize the type of metadata collected at **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.



Set limits

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, etc.
- `cribl`: Reports the Cribl Edge version, mode, Fleet for managed instances, 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.

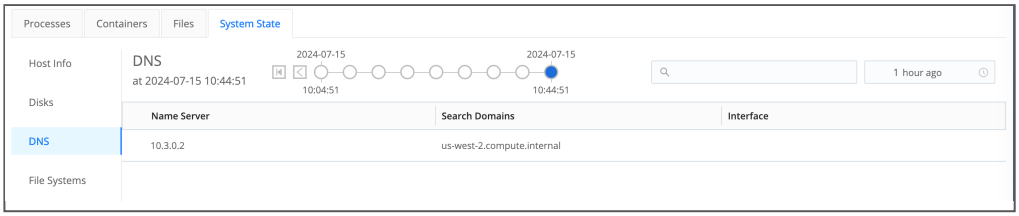
Disks

The **Disks** tab displays the inventory of physical disks and their partitions on the host system.

Disks tab

DNS

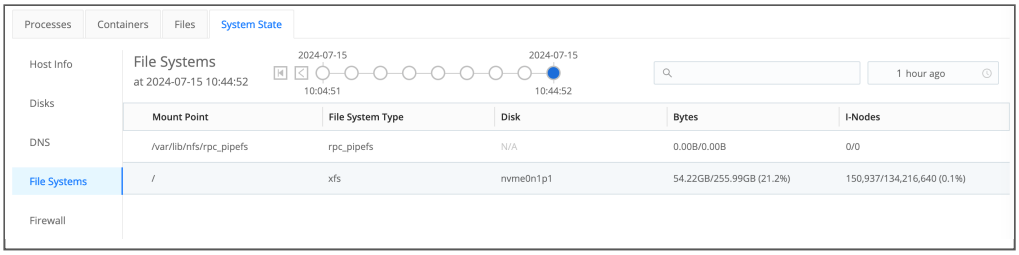
The **DNS** tab lists the host system's DNS resolvers and search entries.



DNS tab

File Systems

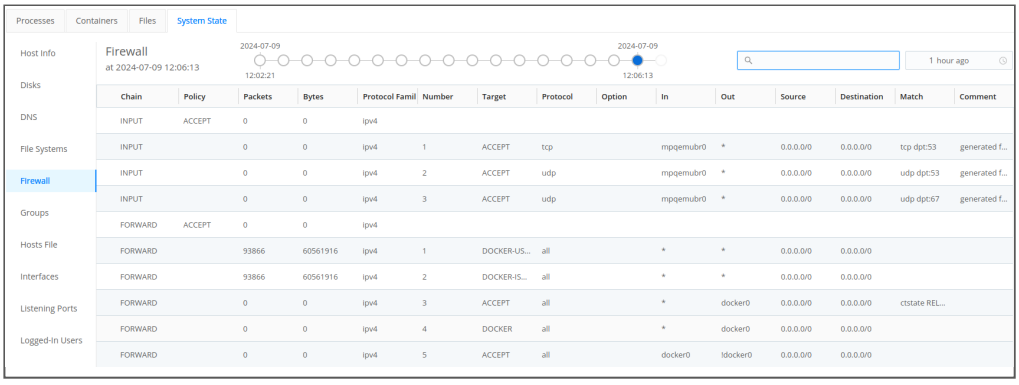
The **File Systems** tab displays an inventory of the mounted file systems on the host system.



File Systems tab

Firewall

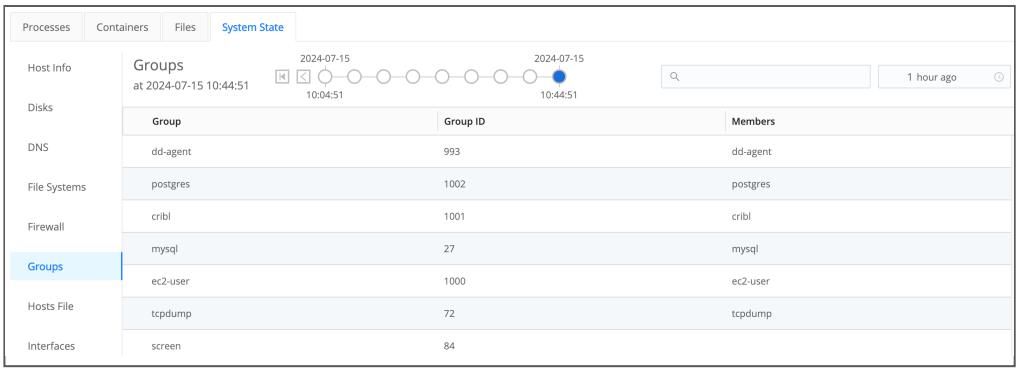
The **Firewall** tab displays a list of the host's defined firewall rules.



Firewall tab

Groups

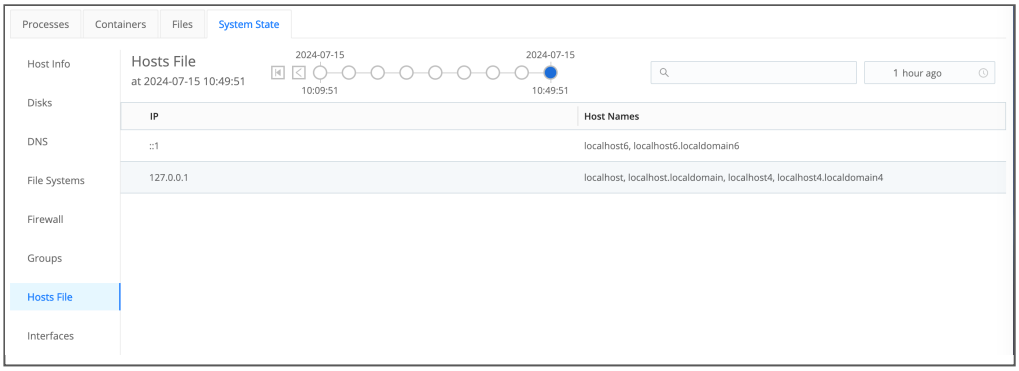
The **Groups** tab displays a list of the local groups including their names, descriptions, and members on the host system.



Groups tab

Hosts File

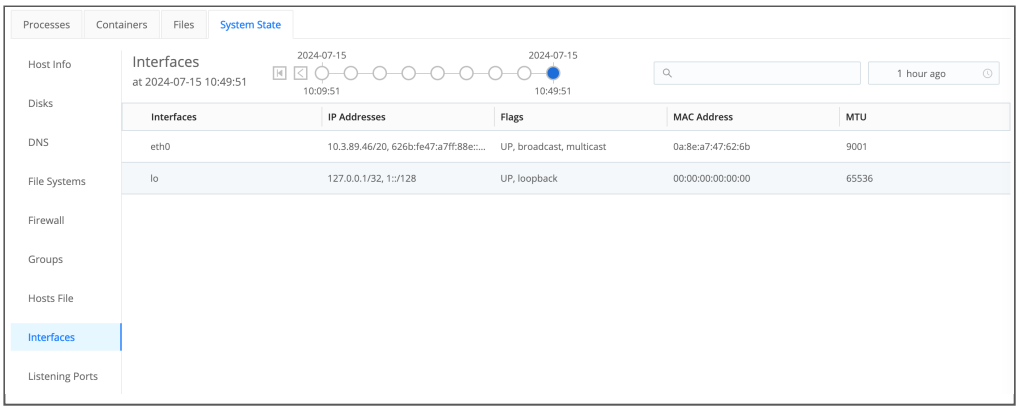
The **Hosts File** tab displays the host system's current state.



Hosts File tab

Interfaces

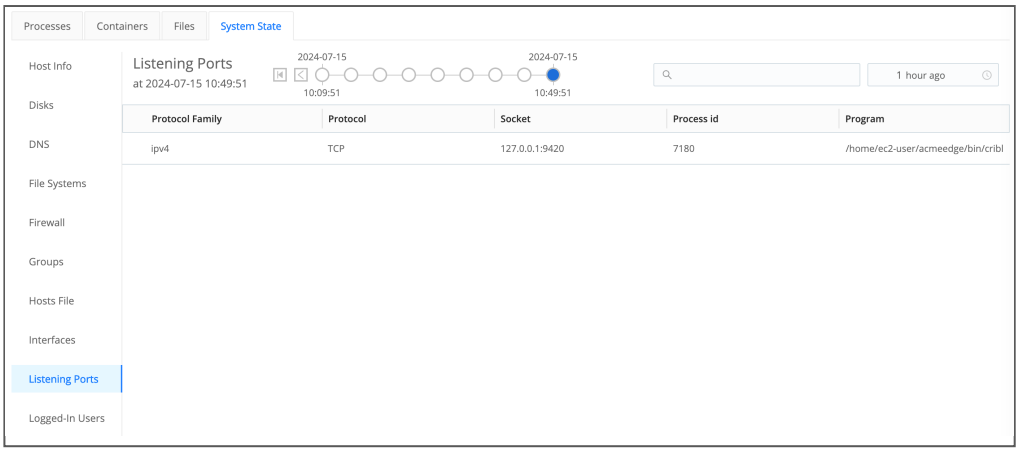
The **Interfaces** tab displays a list of each of the network interfaces on the host system.



Interfaces tab

Listening Ports

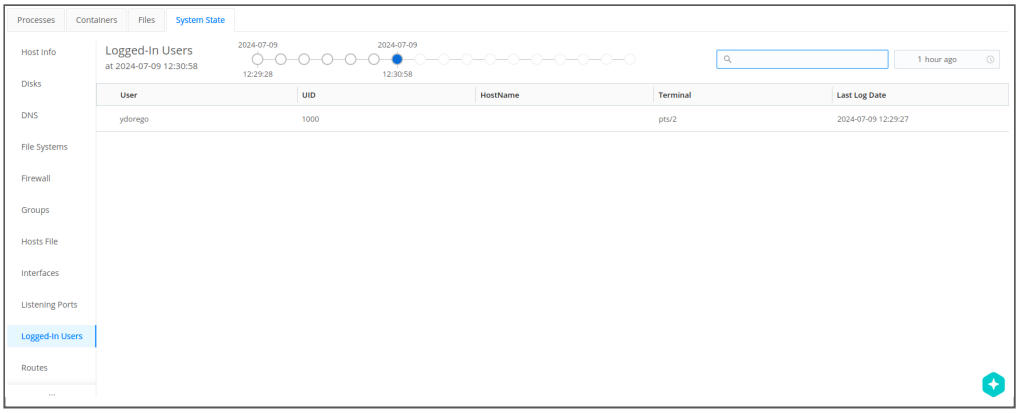
The **Listening Ports** tab displays a list of listening ports and their associated process identifier (pid).



Listening Ports tab

Logged-In Users

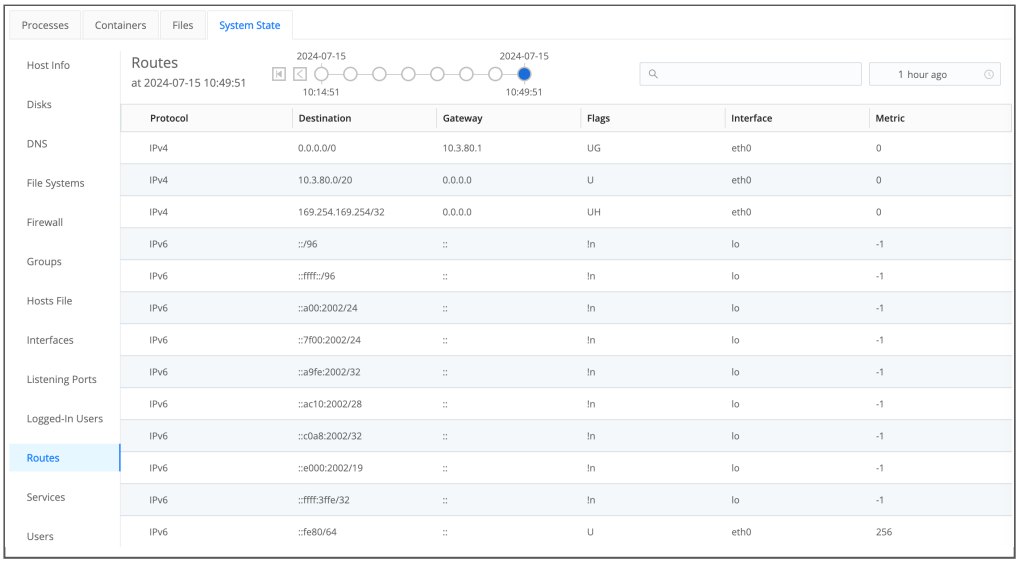
The **Logged-In Users** tab displays a list of currently logged-in users on the host.



Logged-In Users tab

Routes

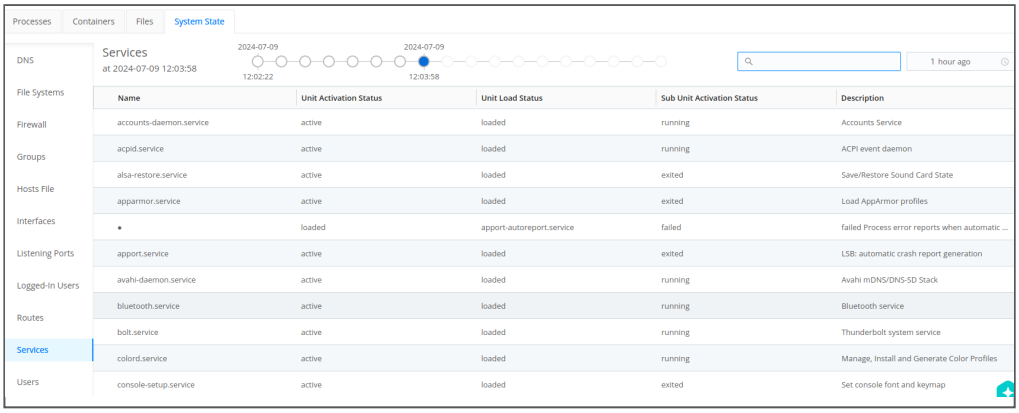
The **Routes** tab displays entries from the network routes on the host system.



Routes tab

Services

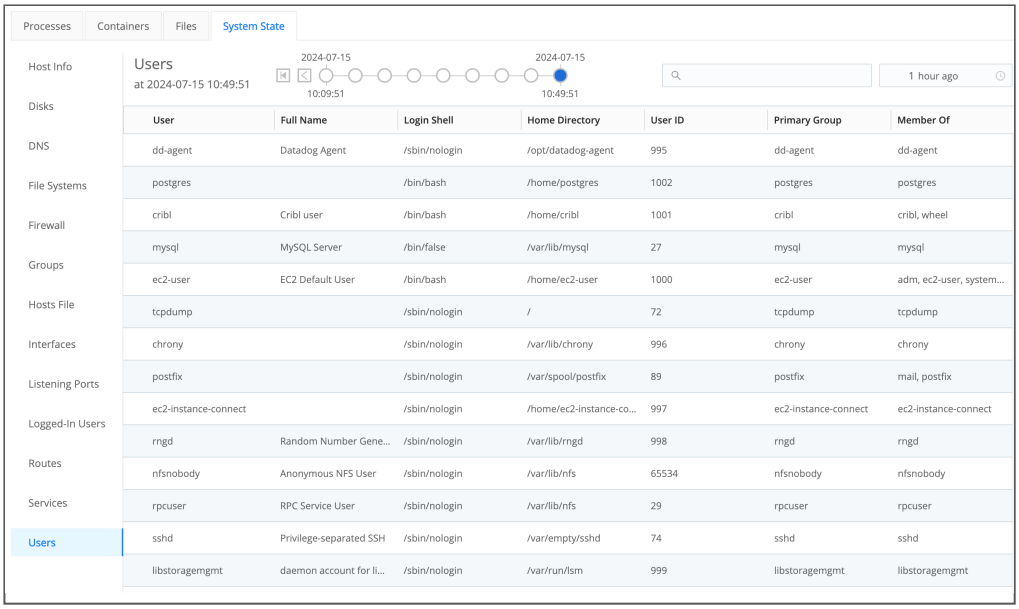
The **Services** tab displays a list of each configured service (such as `systemd` and `initd`) along with their running status.



Services tab

Users

The **Users** tab displays a list of local users on the host system.



Users tab

`__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.

1.7. Explore 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.

Cribl Edge supports these versions of Windows:

Version	Workstation Type	Notes
Windows 10 and 11	Always-on network connection to the Leader.	Support for Windows workstations that do not have power management features enabled (sleep or hibernate).
Windows 10 and 11	**(Preview Feature)** Laptop or desktop with power management enabled.	Keep these things in mind when deploying Cribl Edge on Windows 10 and 11 workstations that are **not** always on: • 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. • Excessive logging has been observed when a Leader is unavailable to receive heartbeat messages from an Edge Node.
Windows Server 2016	Any	N/A
Windows Server 2019	Any	N/A
Windows Server 2022	Any	N/A

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 AppScope and System Metrics Sources are unavailable on Windows.
- 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.

Access Cribl Edge on Windows

When you first log into Cribl Edge on Windows, you'll land on the **Edge Home** page. From here, select **Edge Nodes** and open the link to a specific Edge Node to teleport into it. See Teleport into an Edge Node for more details on this process.

Now that you're teleported into a specific Node, you can view the metrics and log data that the Node has auto-discovered, and can manually discover and explore other data of interest.

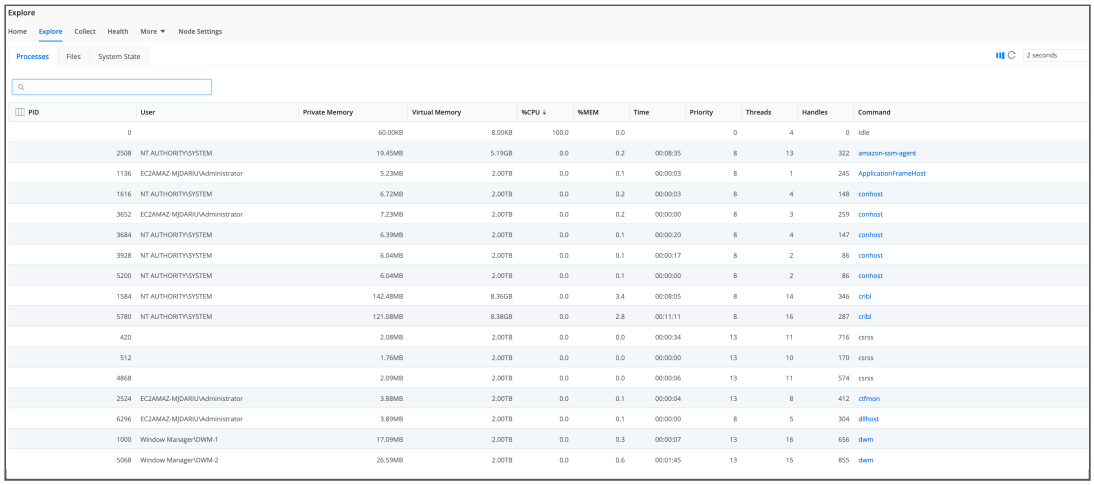
Edge on Windows

From the **Explore** page, you can view more details on your node via the following tabs:

- Processes
- Files
- Host Info/Metadata

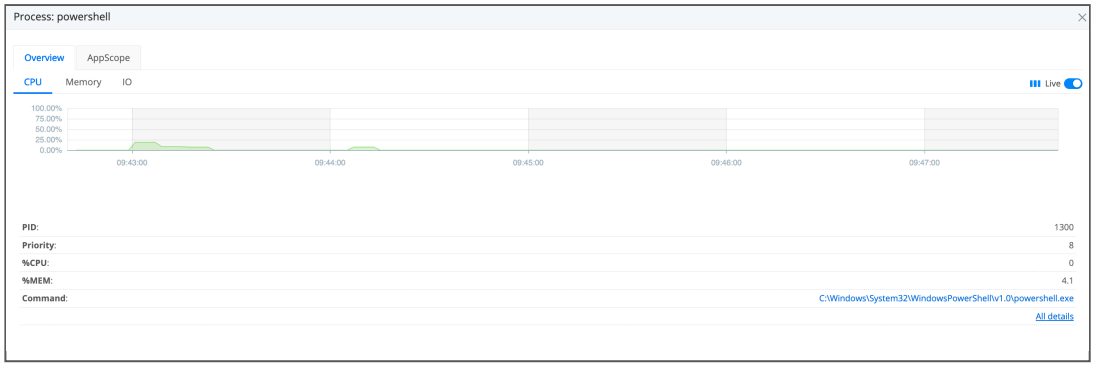
Processes

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



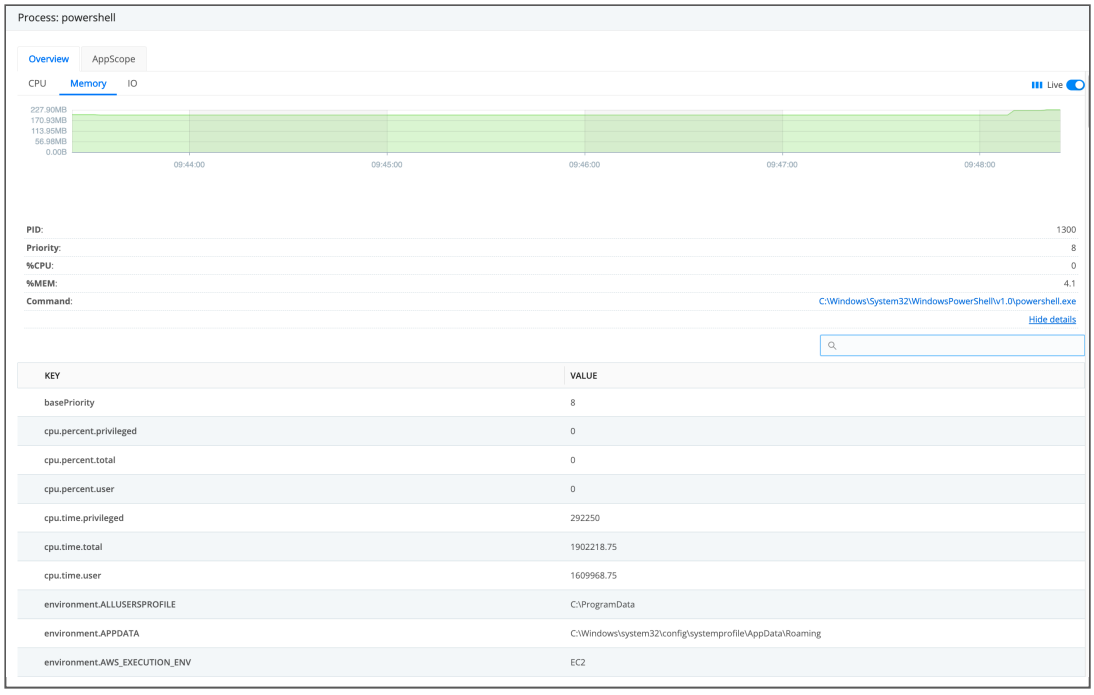
Processes tab

Click on any of the rows to display a modal with basic information on the process, including CPU usage, Memory usage, and I/O graphs.



Process details

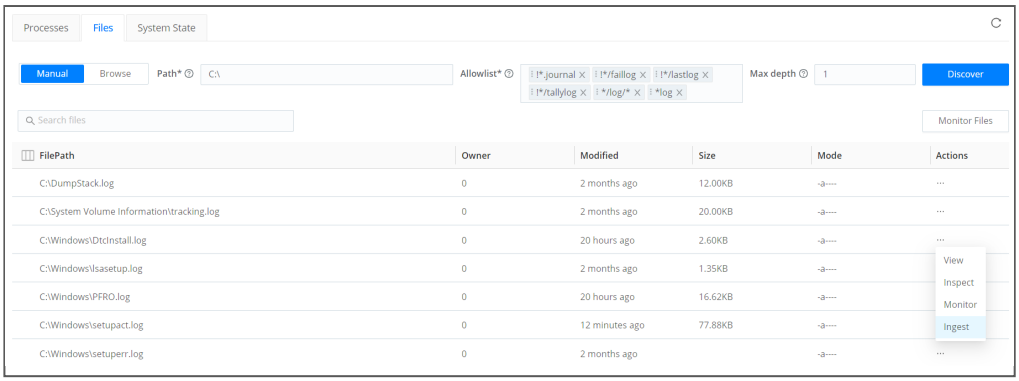
In this modal, click **All details** to get a table view of processes' information. To access this information, you would normally need to SSH to the machine; this view makes troubleshooting across multiple systems much easier.



All detail link

Files

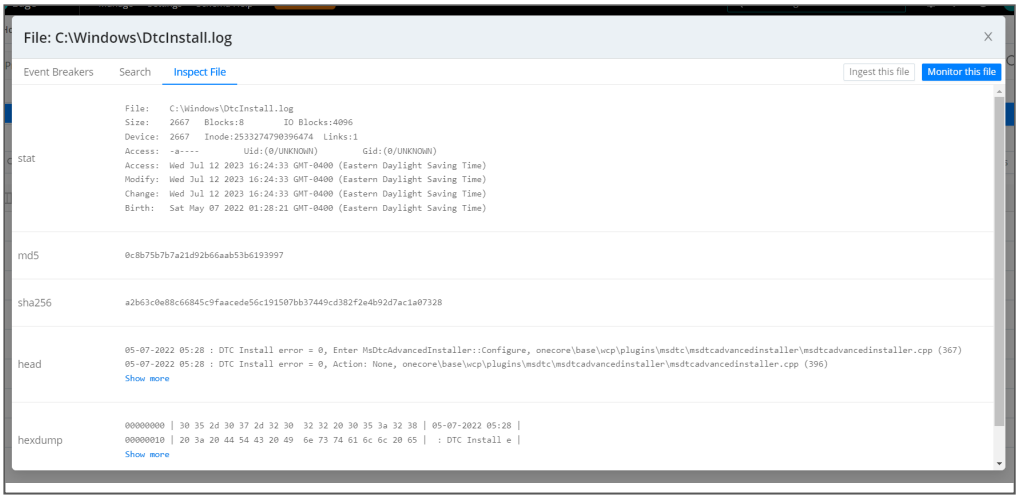
The **Files** tab enables you to specify a list of directories and files to actively monitor.



File tab -Manual mode

The **Actions** column allows you to:

- **View**: Displays a representation of the lines this column contains. You can also click 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.



Inspect File tab

- **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

There are two discovery modes:

- **Manual**

- **Browse**

The **Manual** mode provides the following options:

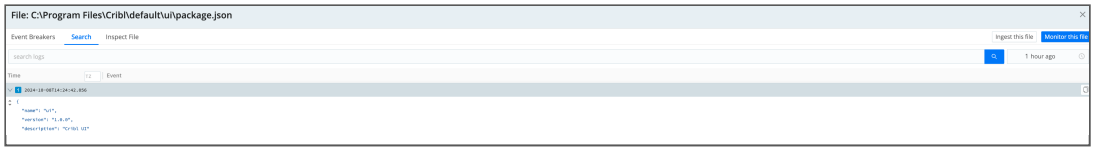
Path

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

Allowlist

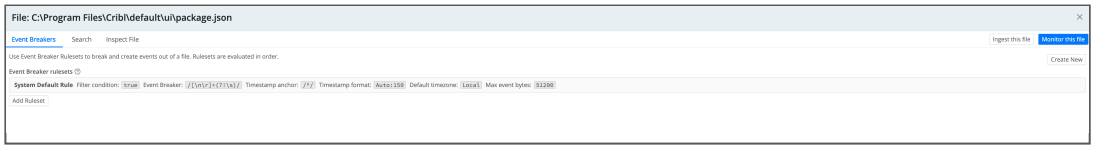
The **Allowlist** field supports wildcard syntax, and supports the exclamation mark (!) for negation. For example, you can use `!*cribl*access.log` to prevent Cribl Edge from discovering its own access log. The default filters are `*/log/*` and `*log`.

Click 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.



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.



Event Breakers tab

Max Depth

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

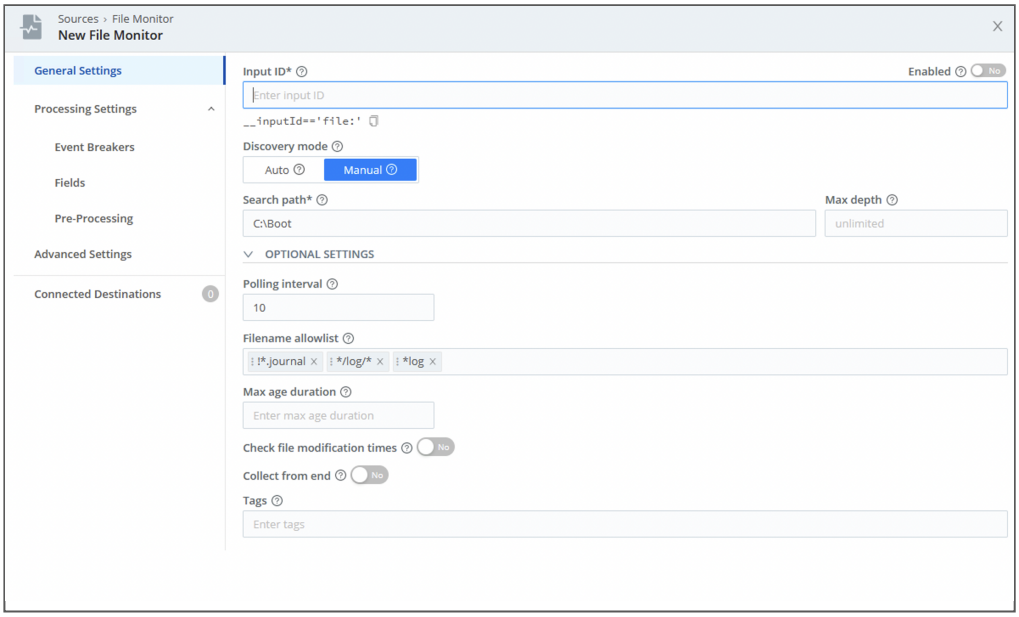
If you enter `0`, Cribl Edge will discover only the top-level files within the specified path. If you specify `1`, Cribl Edge will discover files one level down from the top. Follow this pattern to specify the depth you want.

Monitor Files

The **Monitor Files** button opens a **New File Monitor** modal prefilled with the discovery mode, path, max depth, and allowlist entries specified on the **Files** tab.

Monitor

Click a file's **Monitor** button to configure your File Monitor Source to generate events from the file's lines or records.

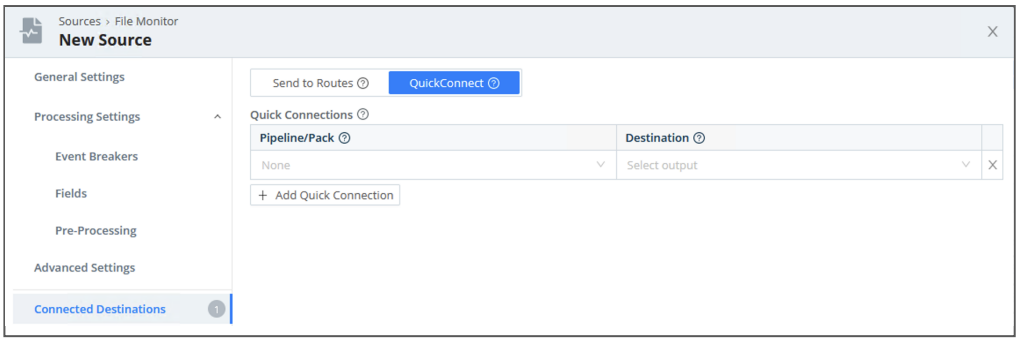


New File Monitor modal

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

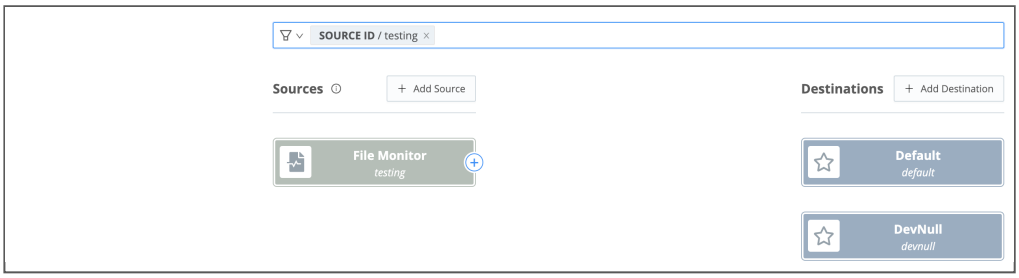
- **Search path**
- **Discovery mode**: Defaults to **Manual** for Windows.
- **Max depth**
- **Filename allowlist**

Note that the **Connected Destinations** section defaults to **QuickConnect**, Cribl Edge's graphical UI. 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.



Connected Destinations

For details about making these connections, see File Monitor and QuickConnect.



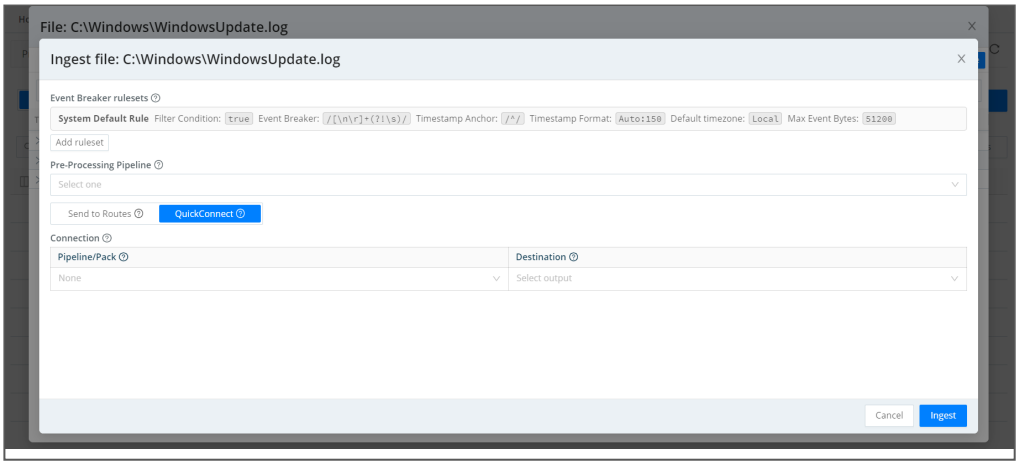
Collect page

Ingesting a File

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

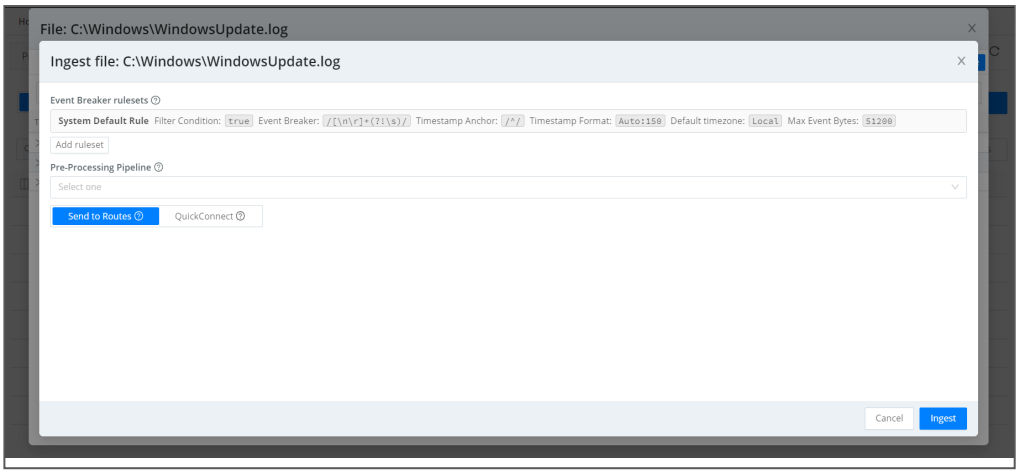
You have two options:

- Send directly to a configured Destination via **QuickConnect** (the default).



Ingest file via QuickConnect

- Send to Routes through (an optional) Pre-Processing Pipeline.



Ingest file via Routes

You can configure Event Breakers and rulesets for both options.

Exploring Files with Event Breakers

When you click a file in the **Files** tab, and Cribl Edge shows a representation of the lines that the file contains – how does that work? Cribl Edge is applying a default Event Breaker to format 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, click **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 these left tabs:

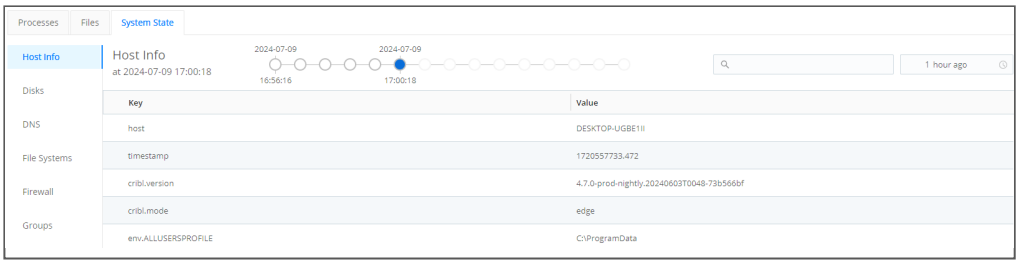
- Host Info
- Disks
- DNS
- File Systems
- Firewall

- Groups
- Hosts File
- Interfaces
- Listening Ports
- Logged-In Users
- Routes
- Services
- Users

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

Host Info/Metadata

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



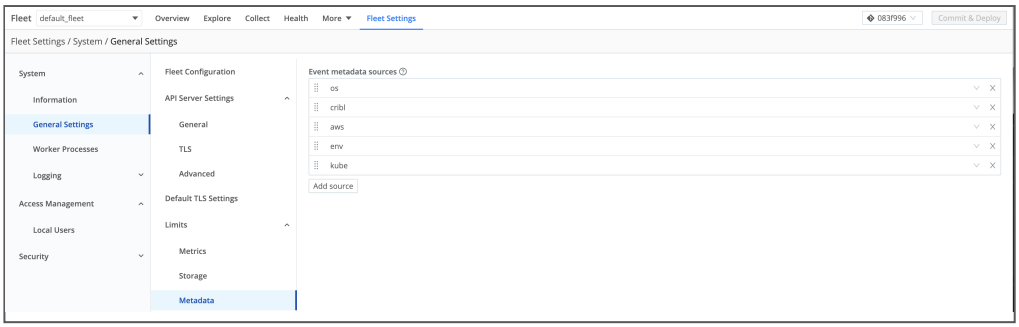
Host info / Metadata collected

The metadata surfaced by an Edge Node can be used 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 **Settings** > **General 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.



Set limits

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, etc.
- `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.

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.

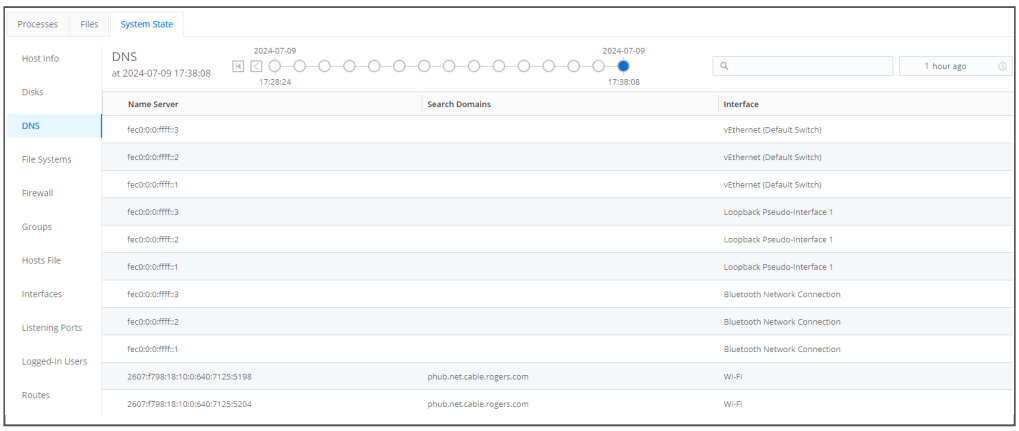
Disks

The **Disks** tab displays the inventory of physical disks and their partitions on the host system.

Disks tab

DNS

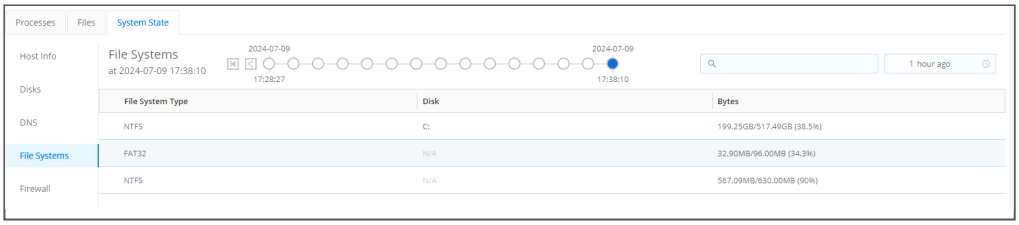
The **DNS** tab lists the DNS resolvers and search entries on the host system.



DNS tab

File Systems

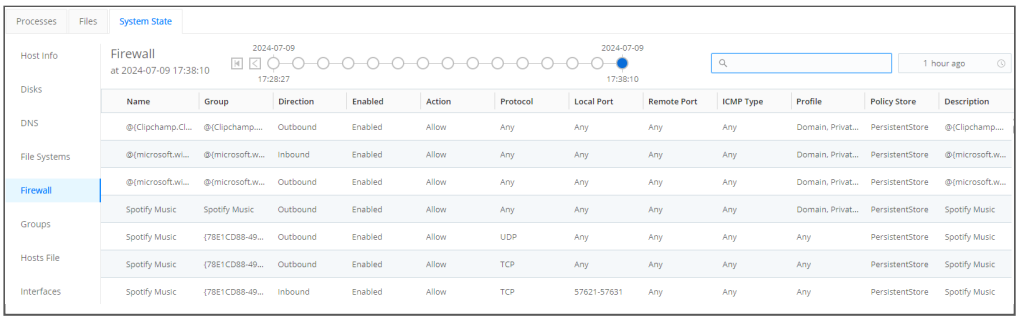
The **File Systems** tab displays an inventory of the mounted file systems on the host system.



File Systems tab

Firewall

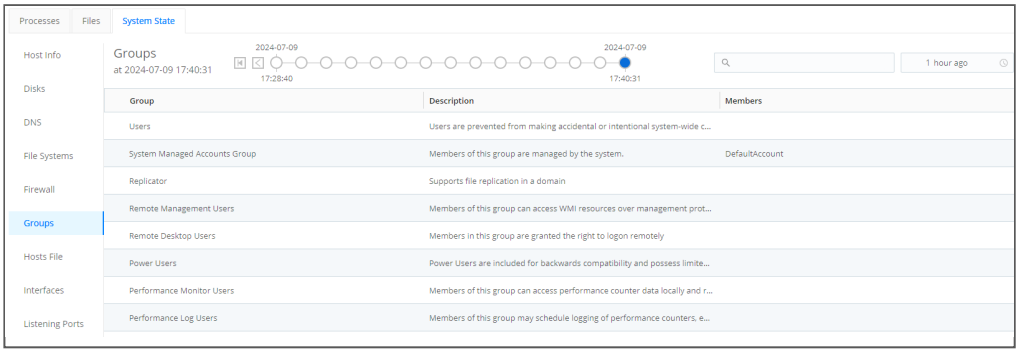
The **Firewall** tab displays a list of the host's defined firewall rules.



Firewall tab

Groups

The **Groups** tab displays a list of local groups including their names, descriptions, and members on the host system.



Groups tab

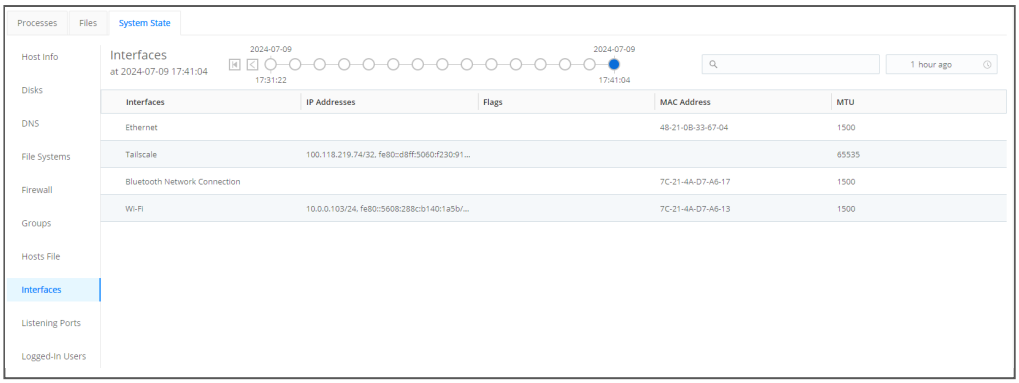
Hosts File

The **Hosts File** tab displays the current state on the host system.

Hosts File tab

Interfaces

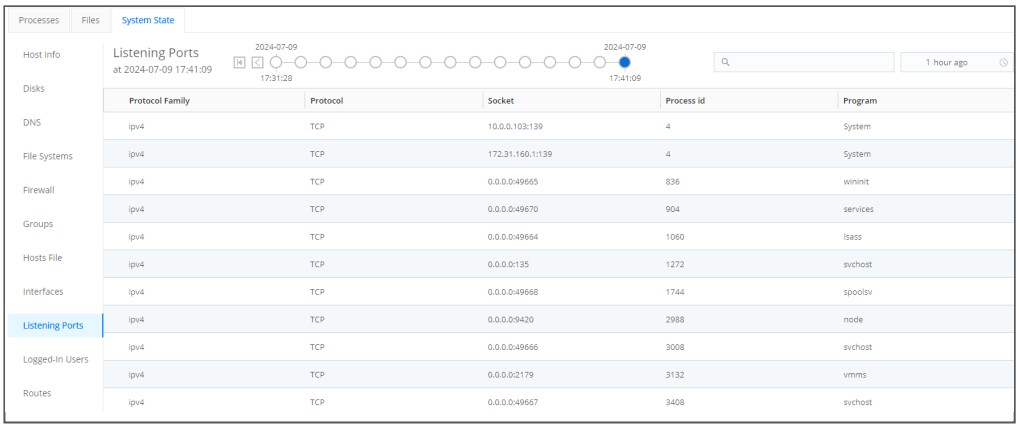
The **Interfaces** tab displays a list of each of the network interfaces on the host system.



Interfaces tab

Listening Ports

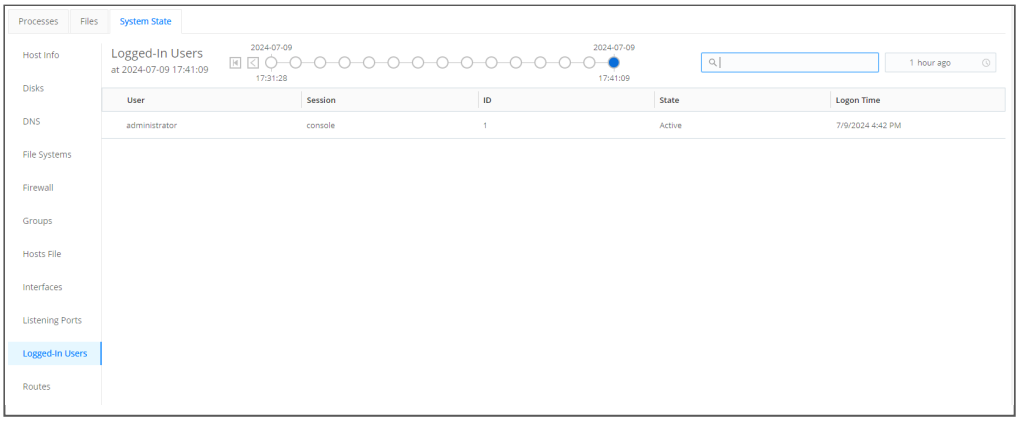
The **Listening Ports** tab displays a list of listening ports and their associated process identifier (pid).



Listening Ports tab

Logged-In Users

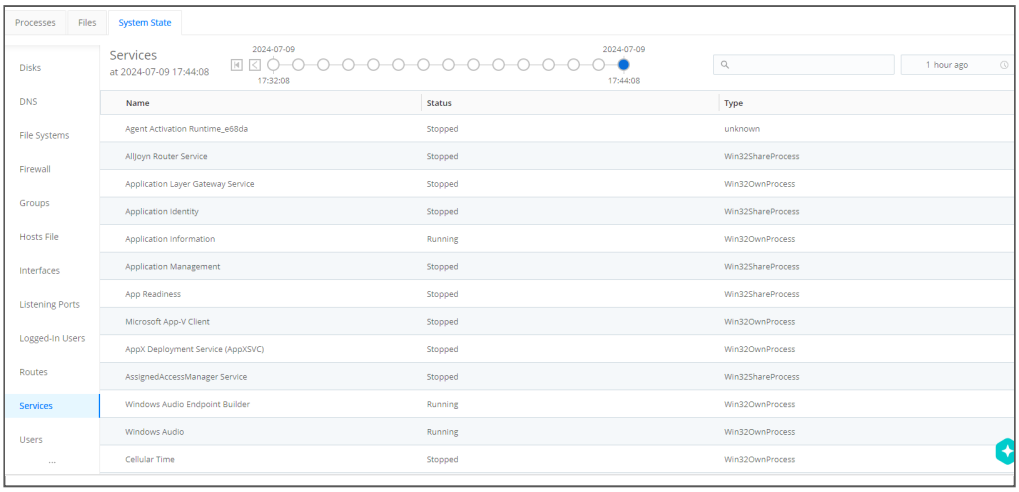
The **Logged-In Users** tab displays a list of currently logged-in users on the host.



Logged-In Users tab

Services

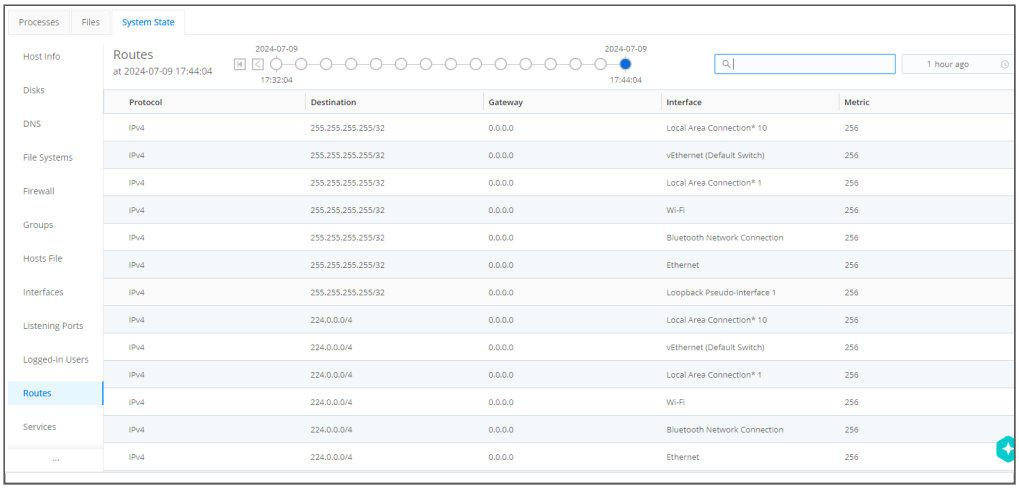
The **Services** tab displays a list of each configured service along with their running status.



Services tab

Routes

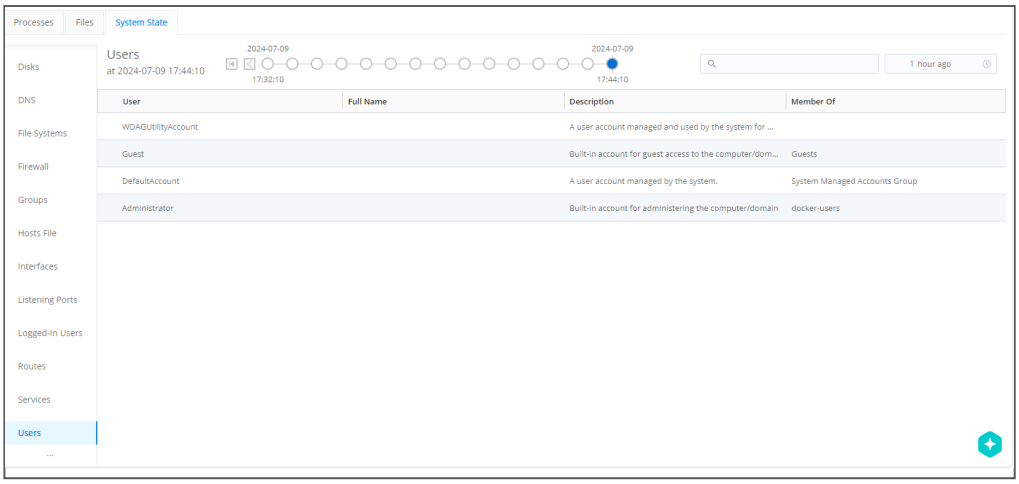
The **Routes** tab displays entries from network routes on the host system.



Routes tab

Users

The **Users** tab displays a list of local users on the host system.


Processes
Files
System State
Disks
DNS
File Systems
Firewall
Groups
Hosts File
Interfaces
Listening Ports
Logged-In Users
Routes
Services
Users
...
Users
at 2024-07-09 17:44:10
2024-07-09
17:32:10
2024-07-09
17:44:10
1 hour ago
User
Full Name
Description
Member Of
WDAGUtilityAccount
A user account managed and used by the system for ...
Guest
Built-in account for guest access to the computer/dom...
Guests
DefaultAccount
A user account managed by the system.
System Managed Accounts Group
Administrator
Built-in account for administering the computer/domain
docker-users

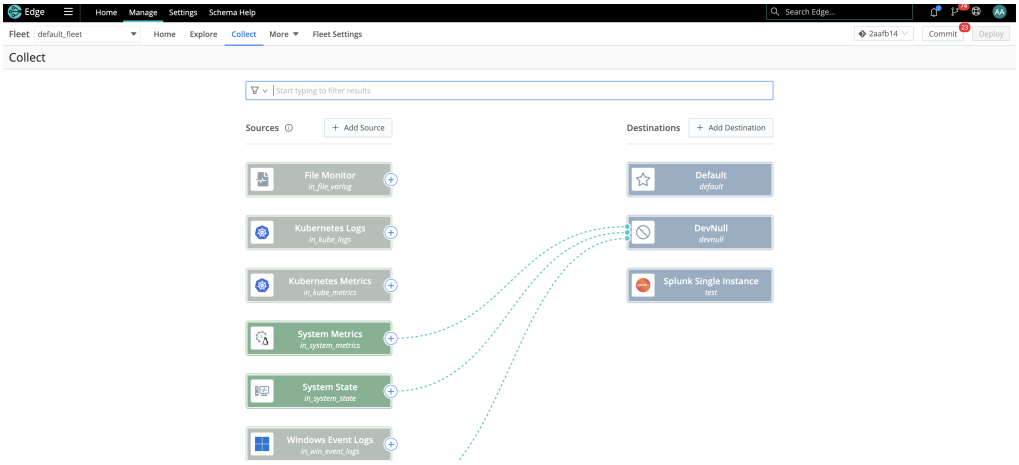
Users tab

1.8. QuickConnect

The QuickConnect visual 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 Cribl Edge's full range of data-transformation [Functions] (functions). 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**, click **Collect**.

To display QuickConnect in a distributed deployment, you must first select a **Fleet**. Click 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**.

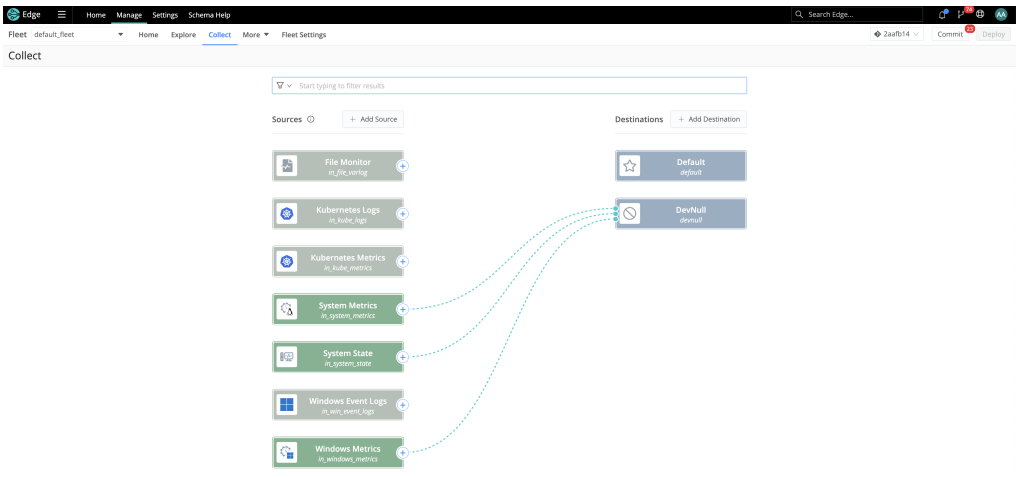
QuickConnect Versus Routes

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's options to clone and cascade data across Pipelines.
- Most Sources are available to fully configure, with a few current exclusions. You'll need to use **Routing** > **Data Routes** to configure Collector Sources and these four Collector-based Sources:
    - Prometheus Scraper
    - Office 365 Activity
    - Office 365 Services
    - Office 365 Message Trace
- 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:

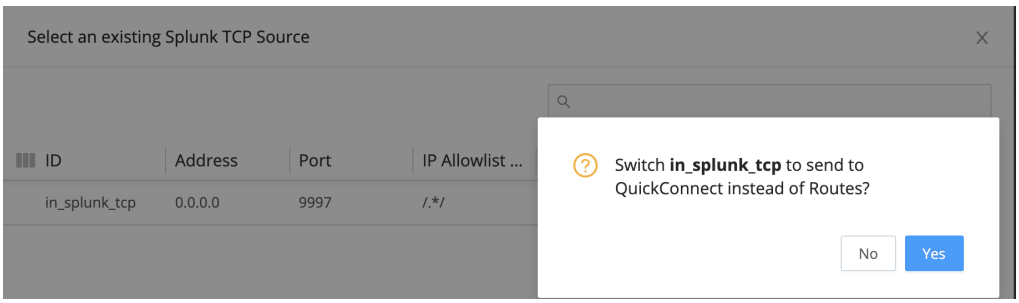


Enabling Sources and Destinations

When you click to add 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 click **Save**.

Switching Contexts

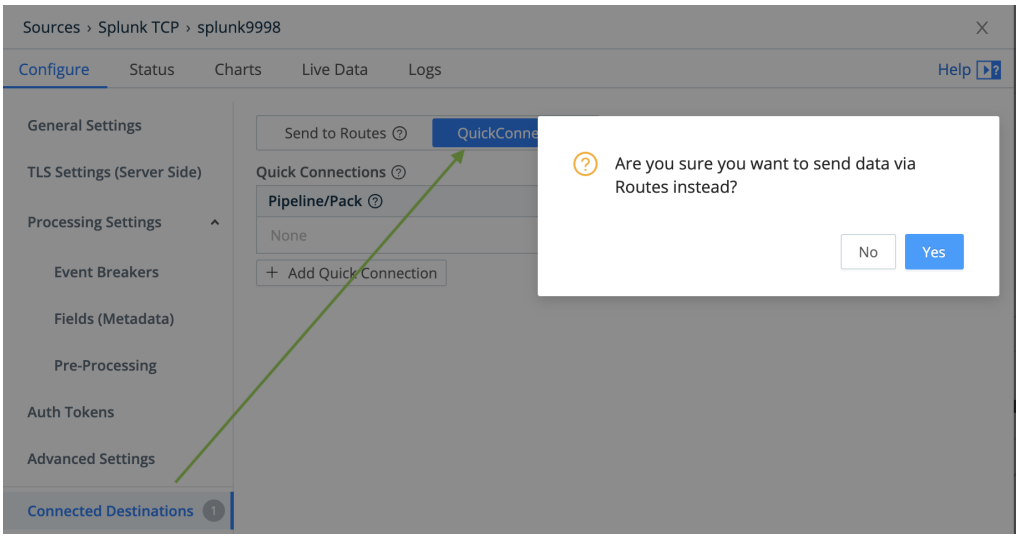
If you switch an **existing** Source to QuickConnect – for example, one of the preconfigured Sources that ship with Cribl Edge – you'll see the dialog shown below. This is due to QuickConnect's independence from the Routes interface – you need to confirm your choice to move the existing configuration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

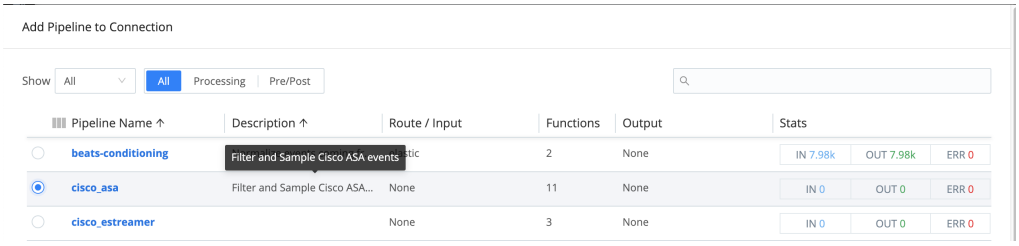
You can also go the opposite way: move a Source originally configured in QuickConnect to the Routing interface. In the Source's drawer or config modal, select the **Connected Destinations** left tab, and then click the resulting tab's **Send to Routes** button. Here again, you'll need to confirm your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



Switching a Source/Destination between worlds

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

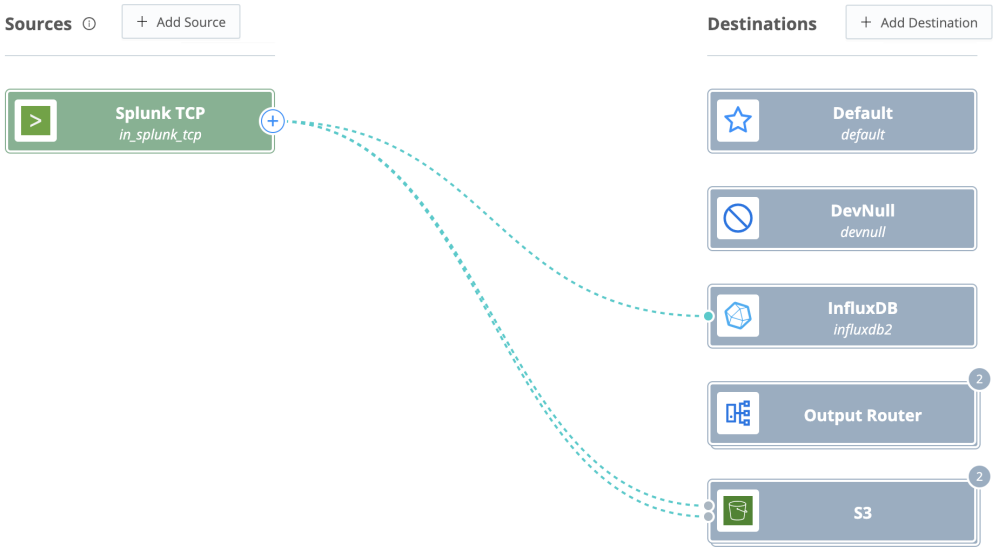
Here, you're selecting among already-configured Pipelines or Packs. **Resist the temptation to click on a bold blue Pipeline or Pack name.** (Doing so will open the Pipeline's or Pack's config in a separate browser tab, which might offer more configuration than you bargained for.)

Instead, click the subtle radio button to the left of the Pipeline or Pack you want to select. (Or click on some black text around the center of your desired Pipeline's/Pack's row, which will fill in its radio button.) Then click **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:



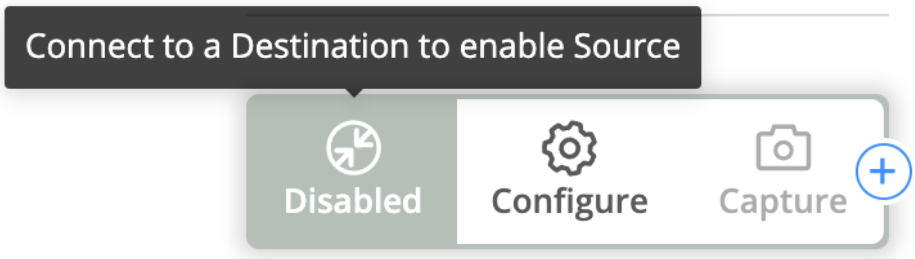
Multiple connections out, multiple connections in

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

Clicking the middle **Configure** button reopens the Source's Destination's configuration drawer, where you can address any problems or simply update the configuration.

The right **Capture** button captures a sample of data flowing through the Source or Destination, in a drawered version of the Source's/Destinations's config modal > [Live tab]](data-preview#capturing-from-a-single-source-or-destination). (Not to be confused with a healthy tile's left **Live** indicator.)

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.

4 similar Destinations, stacked

The number at the stack's upper right shows how many tiles it contains.

Hovering over a stack doesn't change its options...until you click on it. This will expand all the tiles in the group, so each tile can now reveal the on-hover options described above.

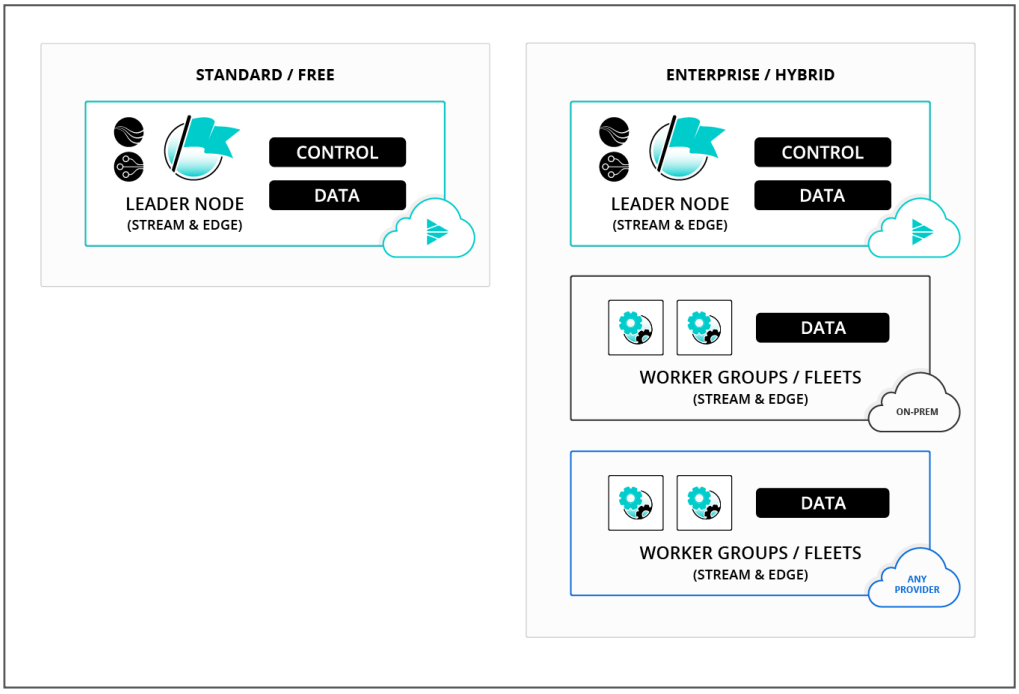
You can drag multiple connection lines from, or to, the same stack. If you drag or modify a connection from or to a stacked group, the UI will prompt you whether to similarly expand the group to access individual tiles.

2. Cribl.Cloud

If you select a Cribl.Cloud deployment, Cribl assumes responsibility for hosting and managing your Cribl Edge infrastructure.

Cribl.Cloud comes in both free and paid versions. 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 an overview of additional features available on Enterprise plans, see Pricing.

Why Use Cloud Deployment?

Cribl.Cloud is designed to simplify deployment, and to provide certain advantages over using your own infrastructure, in exchange for some current restrictions (because Cribl will manage some configuration on your behalf). Cribl.Cloud offers these advantages, among others:

- Full Cribl Edge power with no responsibility to install or manage software for the Leader Node. Cribl.Cloud is fully hosted and managed by Cribl, so you can launch a configured instance within minutes.
- Access to Cribl Search to search, explore, and analyze machine data in place.

- Access to Cribl Lake, a data lake solution for long-term, full-fidelity data storage.
- Access to Workspaces, where you can isolate parallel Cribl Leaders with separate access controls and other configurations.
- Automated delivery of upgrades and new features.
- Leaders and Cribl-managed Edge Nodes encrypt data at rest (including configuration and sample files) at the disk level.
- Free, up to 1 TB/day of data throughput (data ingress + egress) for all new accounts.
- Quick expansion of your Cribl.Cloud deployment beyond the free tier's limits by purchasing credits toward metered billing. Pay only for what you use.

For a more-detailed comparison of Cribl.Cloud versus customer-managed ("on-prem") deployments of Cribl Edge software on your own infrastructure, see Cribl.Cloud vs. Self-Hosted.

If you're new to Cribl Edge, please see our Basic Concepts page and Getting Started Guide for orientation. Cribl.Cloud always runs in Distributed mode – for details about this, see Simplified Distributed Architecture.

Cloud Pricing

Beyond the free tier, an optional paid Cribl.Cloud account – whether Standard or Enterprise – offers direct support, plus expanded daily data throughput according to your needs. From your Cribl.Cloud Organization, select **Go Enterprise** to submit an inquiry about upgrading your free account, and Cribl will respond.

You'll pay only for what you use – the data you send to Cribl Edge, and the data sent to external destinations. However, data sent to your AWS S3 storage is always free. For details, see Pricing.

2.1. Initial Cribl.Cloud Setup

Your first step to using Cribl.Cloud is to sign up on the Cribl.Cloud portal (see Registering a Cribl.Cloud Portal below), to create your Cribl.Cloud Organization.

Your Organization will display a dedicated portal, which has a network and access boundary that isolates your Cribl resources from all other users. (For details, see Cribl.Cloud Data Isolation.)

The portal will initially be on a **free** Cribl.Cloud plan. Certain throughput and administration limits apply to a free account. When you need more capacity and/or options, it's easy to upgrade to a paid or Enterprise plan – just select **Go Enterprise** in your Cribl.Cloud portal.

The Cribl.Cloud Suite is listed on AWS Marketplace. When you're ready for a paid plan, you can use your Enterprise Discount Program (EDP) credits here to run Cribl products, billed through your AWS account – with no need for a separate procurement process.

Registering a Cribl.Cloud Portal

Ready to take the plunge? Here's how to register and manage a Cribl.Cloud instance.

First, if you haven't already signed up on Cribl.Cloud:

1. Start at: https://cribl.cloud/signup/
2. Register with your work email address.
3. Follow the on-screen instructions to provide your name and additional (optional) personal information, agree to the Subscription Services Agreement, and then use the welcome email you receive from Cribl to verify your account.
4. On the organizations page, select **Add Organization**.
5. In the **New Organization** modal, enter an **Organization Name** (a friendly alias for the randomly generated ID that Cribl will assign to your Organization), and then 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)`
    - `Asia Pacific (Sydney)`

- `Canada (Central)`

Note that each user can register only one active Organization on the free tier. For details, see notes on Member Permissions.

6. Complete the quick survey to let us know how you plan to use Cribl. The goals you select and the Sources and Destinations you choose will be used to offer you personalized guidance as soon as you open your Workspace.

Troubleshooting Resources

Cribl University's Troubleshooting Criblet on How to Log Into Cribl.Cloud walks you through the whole registration and login flow. 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 Troubleshooting Criblets and Advanced Troubleshooting short courses.

2.2. Managing Cribl.Cloud

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 region it is located in, the software version, and license, by selecting **Organization Details** at the bottom of the sidebar.

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

2.3. Cribl.Cloud Billing

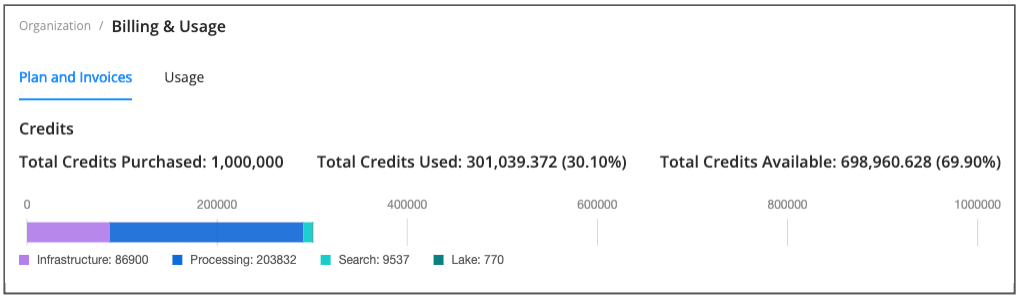
In the Cribl.Cloud portal, you can check the plan of your Cribl.Cloud Organization, track data usage, and preview and download invoices.

Check Cribl.Cloud Plan and Credits

To check the plan that your Organization is running:

1. In the sidebar, select **Organization**, then **Billing & Usage**.
2. On the **Billing & Usage** submenu, select **Plan and Invoices**.

Here you can see a bar of purchased, used, and available **Credits** on your account. Color-coding breaks down usage by infrastructure versus processing (data throughput).



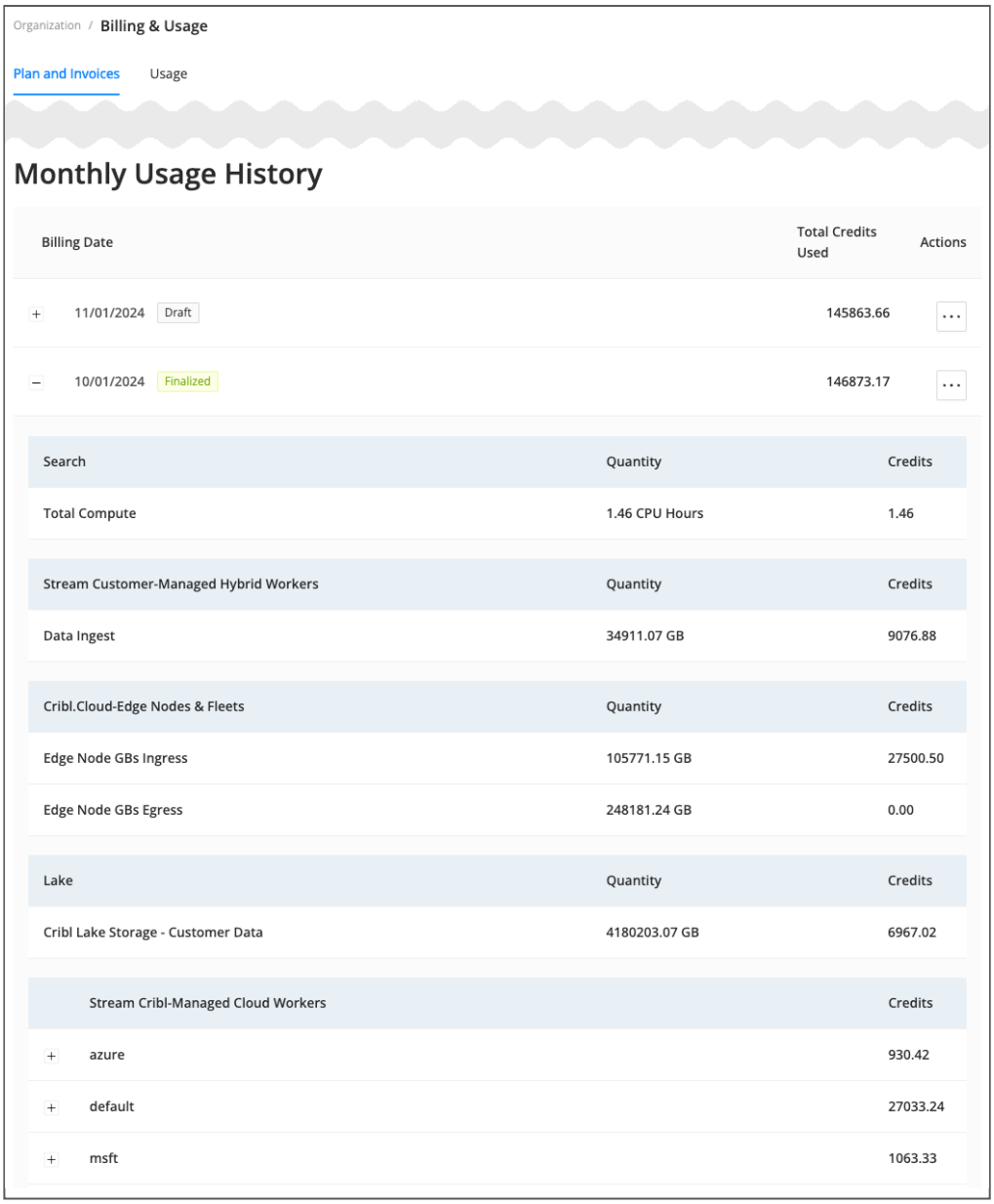
Credit usage bar

Under **Plan**, you can see the details if your Cribl.Cloud plan, including its period and start date.

Check Data History

Monthly Usage History offer details about your credits usage in the current and prior months. Here, you can break out usage on the following:

- Cribl Search
- Cribl Lake
- Hybrid Workers (ingest billing only)
- Groups of Cribl-managed Workers in Cribl.Cloud (with ingest versus infrastructure breakouts per Group)
- Connected Environments GB in (on-prem Leaders passing usage data to Cribl.Cloud). See the article on Connecting On-Premises Leaders to Cribl.Cloud Billing for more information.



Plan and Invoices

Credits carry over across billing periods, as long as you renew your Cribl.Cloud plan.

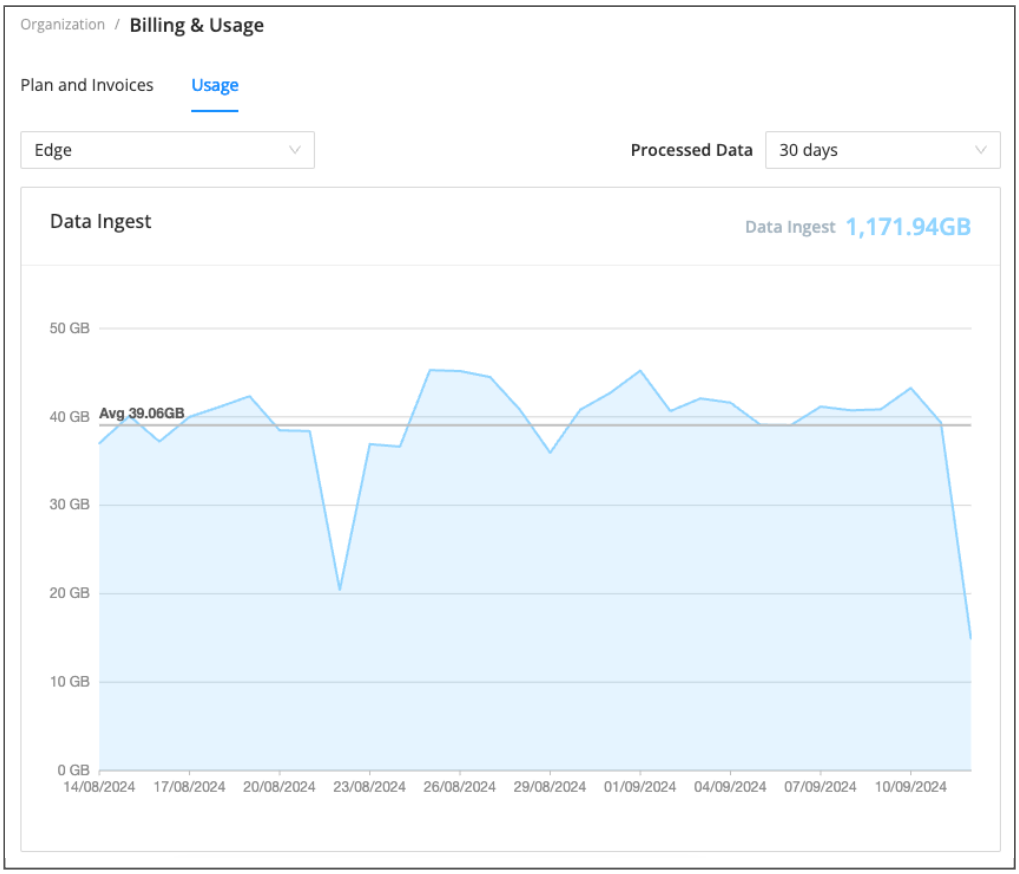
To check detailed information about data usage:

1. In the sidebar, select **Organization**, then **Billing & Usage**.
2. On the **Billing & Usage** submenu, select **Usage**.

Select a product to view its graphs (including any Connected Deployments), and adjust the duration of time for which data was processed over a selectable trailing period of 7 days to 1 year. For more information on Connected Deployments, see Connecting On-Premises Leaders to Cribl.Cloud Billing.

Stream	Search	Lake	Edge
The **Stream** graph shows credits usage for **Data in**, **Data out**, and **Infrastructure**. Using the **Processed Data** drop-down, you can filter the aggregate display down to individual Cribl-managed Groups in Cribl.Cloud, or to all customer-managed (hybrid) Groups.	The **Search** graph shows billed **Search Compute** in CPU hours and **Total Compute Costs**.	The **Lake** graph shows **Total Usage** and **Data Usage**.	The **Edge** graph shows **Data Ingest** and **Data Egress**.

- The trend line shows daily averages, and you can hover over data points to pop out details.
- At the right side of each graph is a total for the selected period.



Billing > Usage tab

Download Invoices

You can download your monthly Cribl.Cloud invoices as CSV or JSON files. Then, use them in external tools to help with budget planning and analytics or store them for historical purposes.

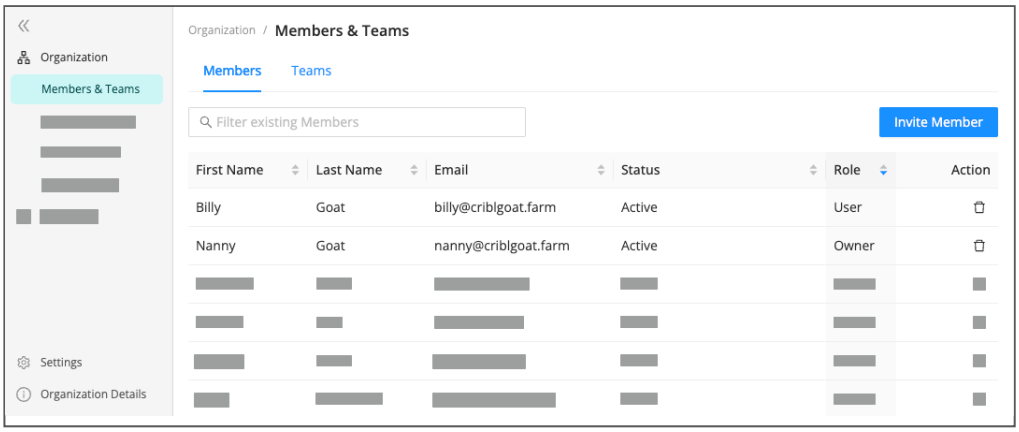
You can only download finalized invoices, not draft ones.

To download an invoice:

1. In the sidebar, select **Organization**, then **Billing & Usage**.
2. On the **Billing & Usage** submenu, select **Plan and Invoices**.
3. Under **Monthly Usage History**, hover over ••• in the **Actions** column of the invoice you want to download.
4. Select **Download CSV** or **Download JSON**.

2.4. Managing Cribl.Cloud Members

From the **Organization** > **Members & Teams** page, an Organization's Owner can invite new users to join the Organization, assign Permissions to new and existing Members, remove pending invites, and remove existing Members.



Members & Teams page: Managing Invites and Members

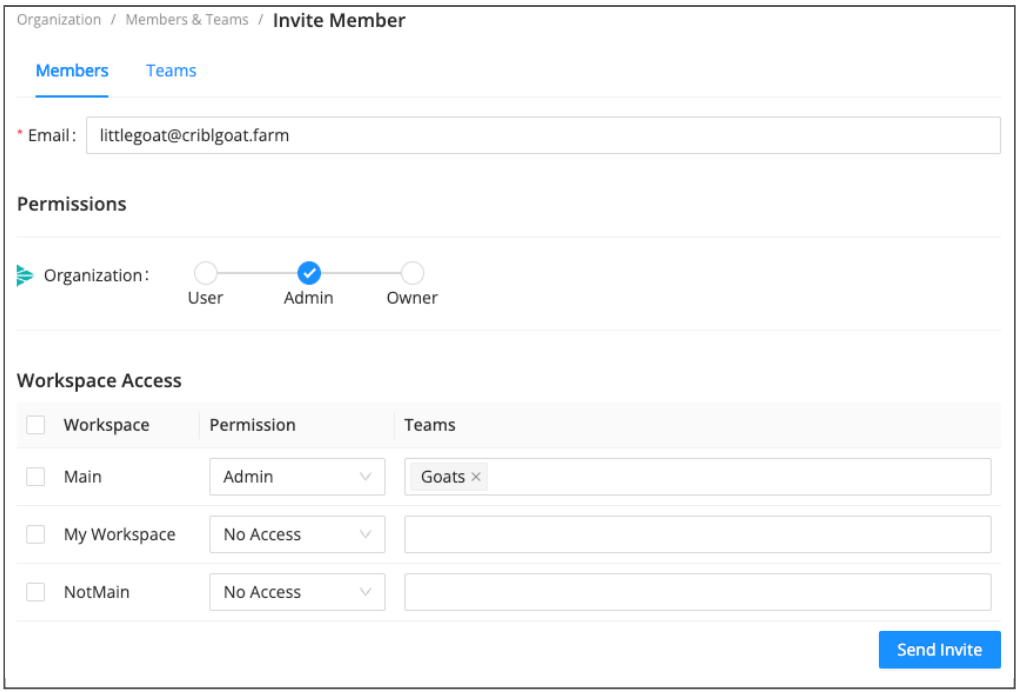
Inviting Members

To invite a new Member:

1. On the top bar, select **Products**, and then select **Cribl**.
2. In the sidebar, select **Organization**, and then select **Members & Teams**.
3. On the **Members & Teams** submenu, select **Members**.
4. Select **Invite Member** to open the page shown below.
5. Enter the **Email** of the new user you want to invite.
6. Under **Permissions**, assign appropriate role on your Organization.
7. If you selected the **User** role, under **Workspace Access**, define access to specific Workspaces.

Assigning the **User** role requires certain plan/license tiers. For details, see Pricing.

8. If you selected the **Member** Permission for any Workspace, additionally select product-level permissions for that Workspace.
9. Select **Send Invite** to send the invitation.



Invite Member modal

Responding to Invites

An invited Member receives an email with an **Accept Invitation** link to either sign into their existing Cribl.Cloud account, or else sign up to create an account and its credentials.

After signing in, they'll have access to your Organization and Cribl Edge instance at the Permission level you've specified.

Organization Member Permissions

You assign Permissions per individual user when you invite them to your Organization, or after they join it. Cribl.Cloud does not currently support globally predefining or assigning group Permissions, as with on-prem Cribl Edge.

For Permissions that you can assign at the lower product, Fleet, and resource levels (all of those only with certain plan/license tiers, see our Members and Permissions topic, which also covers on-prem counterparts to the Cribl.Cloud Organization-level Permissions listed here.

Permission	User	Admin	Owner
Log into the system			
Update own Member profile			

Permission	User	Admin	Owner
View Stream Worker groups you have access to			
`Admin` Access to all Products			
View and execute Leader commits			
View and modify Global Settings			
Manage ACLs (Access Control lists)			
View and update SSO settings			
View Data Sources and Trust Policies			
Manage API Credentials			
View and manage Cribl Suite Global Settings			
View and manage (provision, update, and delete) Stream Worker Groups			
View the Organization's Billing dashboards			
Download invoices			
View Organization Details			
Update Organization Details			
Delete an Organization			
View Organization Members			
Manage (invite, update, delete) Organization Members			

Each user can have Owner Permissions for multiple organizations, and each organization can have multiple users as Owners. However, each user – as defined by their email address – can register only one active Organization, and only if they are not already the Owner of a different Organization.

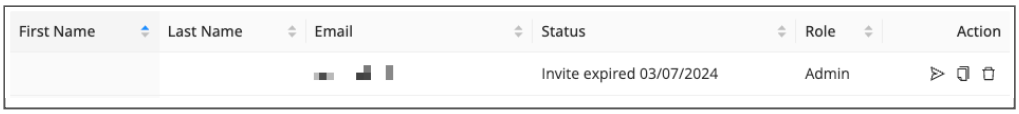
Cribl.Cloud has retired the access-management model of Local Users and Roles/Policies that Cribl applications used prior to v.4.2. Create and configure only **Members** on Cribl applications – references elsewhere in Cribl docs to Local Users do not apply to Cribl.Cloud.

Managing Invites

While an invite is pending, the **Members & Teams** page offers you these options in the **Action** column to deal with commonly encountered issues:

- **Resend Invite**: If your invited Member didn't receive your invitation email, you can click this button to resend it.
- **Copy Invite Link**: If emails aren't getting through at all, click this button to copy and share a URL that will take the invitee directly to the signup page. This target page encapsulates the same identity, Organization, and Permission you specified in the original email invite.
- **Revoke Invite**: This is for scenarios where you need to revoke a pending invite. (You sent someone a duplicate invite, your invitee is spending too much time in space to be a productive collaborator, etc.) After clicking this button, you'll see a confirmation dialog.

After seven days, if an invite has been neither accepted nor revoked, it expires.



Managing Invites

Removing Members

To remove a Member from your Organization:

1. In the sidebar, select **Organization**, and then select **Members & Teams**.
2. Select **Members**, and next to the Member you want to remove, select **Remove Member** from the **Action** column.

Users will retain access to any other Cribl.Cloud Organizations they own or are Members of.

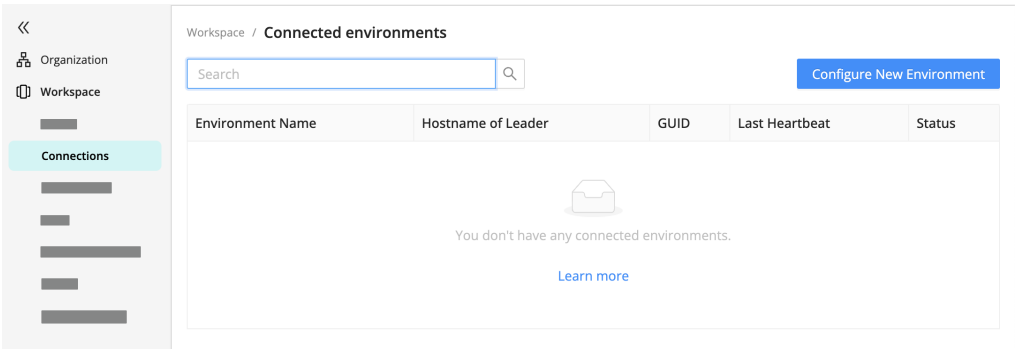
Removing SSO Members

If your Organization is using SSO integration, the IDP (identity provider) admin manages Members in the IDP system. When you remove such a Member from the IDP, they will still be left in the Members list in the UI.

After removing the Member from the IDP, you also need to manually delete them from the Members list in the Cribl.Cloud Portal to completely remove them from the UI.

2.5. Connect On-Premises Leaders To Cribl.Cloud Billing

If you have an on-prem Cribl Edge environment and a Cribl.Cloud account on certain plan/license tiers, you can use the **Connected Environments** option to establish 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.



The Connected Environments UI

This replaces the on-prem license, and gives you the flexibility to maintain your on-premises deployments without managing multiple licenses and tracking spend across environments.

▶ Cribl Edge Environment Components

Connecting your on-prem Leader to Cribl.Cloud allows you to:

- Monitor credit consumption for each of your on-prem Cribl Edge environments from one user interface.
- Create a connection between the Cribl Edge on-prem Leader and the Cribl.Cloud Leader, which receives billing and usage metrics from the on-prem Leader.

Limitations of the Universal Subscription Feature

Connecting environments has some usage limitations:

- The Universal Subscription feature is available on Cribl.Cloud with certain plan/license tiers. For details, see Pricing.
- Single-instance deployments of Cribl Edge aren't eligible for a Universal Subscription.
- Your Distributed deployment's on-prem Leader must be running Cribl Edge 4.7.x or newer.

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:

Create a Cribl.Cloud Account

To use Connected Environments, you must have a Cribl.Cloud account on a certain plan/license tier. See our documentation on Initial Cribl.Cloud Setup.

Decide How You Want to Connect to the Leader

There are three ways to enable a Cribl.Cloud to on-prem Leader connection:

- From the Cribl.Cloud account in **Connections**, **Connected Environments**.
- From the on-prem Leader in **Distributed Settings**, **Cloud Connection**.
- Using the command line or environment variable, from either the Cribl.Cloud Leader or on-prem Leader.

The setup instructions in this article will cover each of the connection options.

Purchase Cribl.Cloud Enterprise Plan with Credits

Universal Subscription gives you the option to manage an existing on-premises license with Cribl.Cloud credits.

To do this, contact your sales team, who will help you convert your license to credits.

Add Connected Environments in Cribl.Cloud

To connect existing on-prem Cribl Edge Distributed deployments to Cribl.Cloud, select an on-prem Leader that is running Cribl Edge 4.7.x or newer, then use the following steps.

If you have multiple environments, such as staging and production enviroments 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-premises 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
	◦ CLI
		Environment variable:
		`CRIBL_DEPLOYMENT="Goat Farm West Product`
		`CRIBL_CLOUD_WORKSPACE_URL="tls://a12bcDE`
		CLI:
		`./cribl cloud-workspace`
		`-d "Goat Farm West Production"`
		`-H example.goat-farm-prod.cribl.cloud`
		`-p 4200`
		`-u a12bcDEFgHiJKLmnOpQRstuvw3x4z56`

4. If you're using the CLI option, follow these steps, in order:
   - Click **Copy Script**.
   - Open a command-line window, in the environment whose Leader you want to connect to Cribl.Cloud.
   - `cd` into the `$CRIBL_HOME/bin` subdirectory.
   - Paste the script results onto the command line, and run the script.

5. 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.

Enable the On-Prem Leader Connection to Cribl.Cloud

You can connect a Cribl Edge Leader to an existing Enterprise Cribl.Cloud Organization.

1. In the sidebar, select **Settings**.
2. Navigate to **Global**, then select **Distributed Settings**.
3. Select **Cloud Connection**.
4. Toggle **Join Cribl.Cloud Workspace** to `Yes`.
5. Fill in the required fields:

Field	Description
Connection URL	The **Cribl.Cloud URL** to connect to this Leader. In your Cribl.Cloud Organization, locate this field value on the sidebar's **Workspace** > **Access** tab.
Connection port	The port for the Cribl.Cloud Leader. Default is `4200`.
Connection auth token	The authentication token for the Cribl.Cloud Leader.

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 and the **Licensing** page will not be accessible. To return to using an on-prem license, toggle **Join Cribl.Cloud Workspace** to `No`, select **Save**, and select **Yes** in the confirmation modal to save changes and restart.

Explore Credit Consumption for Connected Environments

Now that you have a successful connection to your Cribl.Cloud Organization, 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 Cloud Billing and Usage page.

View Usage Data

1. In the sidebar, select **Organization**, then **Billing & Usage**.
2. On the **Billing & Usage** submenu, select **Usage**.
3. Select the product drop-down to choose a **Connected Deployment** of Cribl Edge.

On this page, you can view GB in and GB out for the connected environment.

View Credit Consumption

1. In the sidebar, select **Organization**, then **Billing & Usage**.
2. On the **Billing & Usage** submenu, select **Plan and Invoices**.
3. In **Monthly Usage History**, expand the **Billing Date** and scroll to the **Connected Environments** table.

Here, you will see each on-prem environment that is connected to this Cribl.Cloud Organization, and the credits each environment is consuming.

2.6. Enterprise Cloud

With a Cribl.Cloud Enterprise plan, you have the same options and flexibility that an Enterprise license provides for an on-prem Distributed deployment – and more:

- 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.

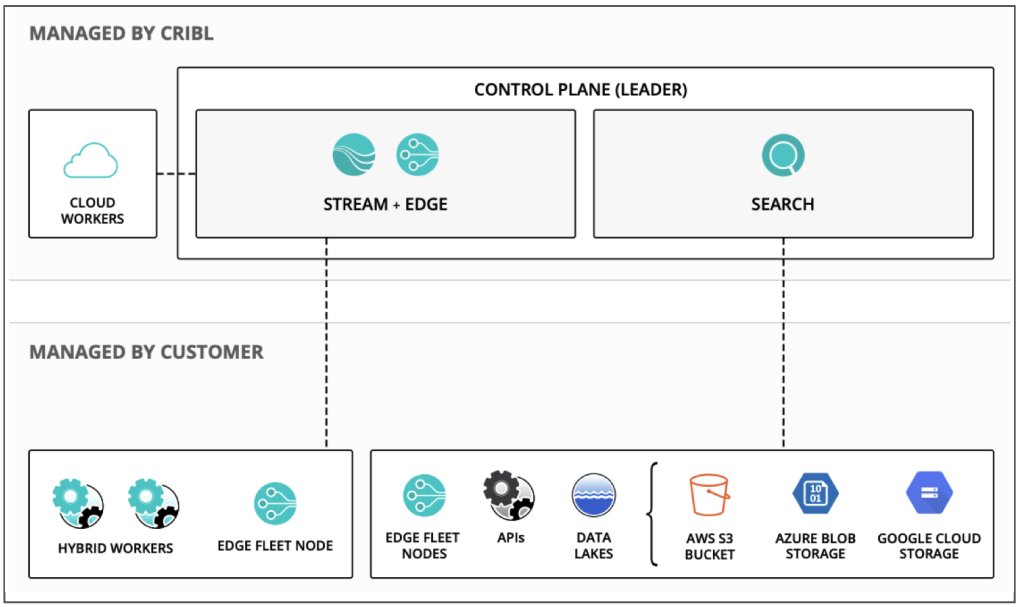
With an Enterprise Plan, the Leader resides in Cribl.Cloud, and controls a flexible mix of Cribl-managed and/or customer-managed Fleets. Cribl manages the Leader's high availability on your behalf.

For other Enterprise features – and for comparisons between Cribl.Cloud plans and on-prem licenses – see Cribl's Pricing page.

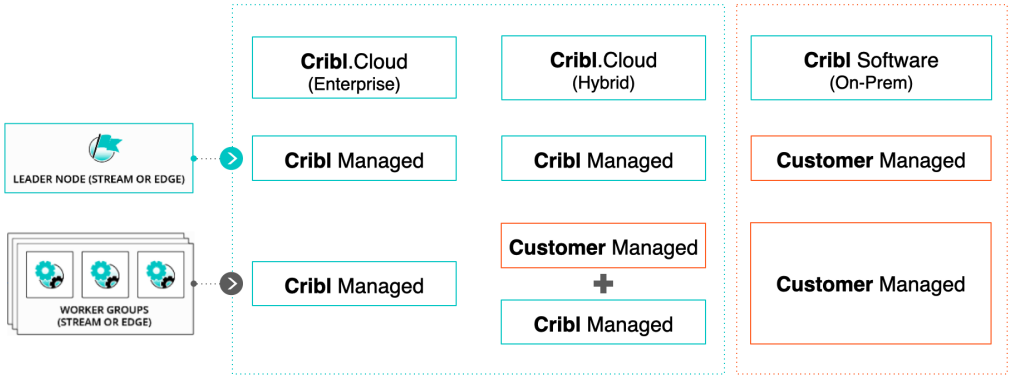
Hybrid Deployment

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-premises 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 deployments, the Leader's Address format is `main-<your-Org-ID>.cribl.cloud`. When configuring a hybrid Worker, use that format in the **Address** field.

To configure hybrid Workers:

1. Log directly into their UI, then select **Settings** > **Global** > **Distributed Settings**. Make sure the **Mode** is set to **Stream: Managed Worker (managed by Leader)** or **Edge: Managed Edge (managed by Leader)** (which might require a restart).
2. Then select **Leader Settings**, and ensure a consistent entry in the **Address** field.

3. Deploying Edge

3.1. Deployment Planning

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-premises 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).

Types of Deployment

- **Cribl Cloud**: Quickly launch a Cribl-managed deployment of the combined Cribl applications suite (Edge, Stream, and Search). In Cribl.Cloud, Cribl provision 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.

System Requirements

Edge Nodes should have sufficient CPU, RAM, network, and storage capacity to handle your **specific** workload. It's very important to test this before deploying to production.

Minimum Requirements for Edge Nodes

Operating System	Minimum Requirements
Linux (RedHat, CentOS, Ubuntu, AWS Linux, Suse)	~1Ghz processor, 512MB RAM, 5GB free disk space (more if persistent queuing is enabled on Edge Nodes)
Windows Server (64-bit), Windows 10 and Windows 11 Workstations (See, Windows Requirements for details)	~1Ghz processor, 512MB RAM, 5GB free disk space

Edge Node sizing depends on specific workload requirements and organizational constraints. While minimum specifications are provided, optimal configurations vary widely. Consider data volume, types, processing needs, and organizational standards when selecting hardware.

Recommended Requirements for the Leader Node

Operating System	Recommended
Linux (RedHat, CentOS, Ubuntu, AWS Linux, Suse)	+4 physical cores, +8GB RAM, 5GB free disk space

Linux Requirements

- 64-bit kernel >= 3.10 and glibc >= 2.17, OpenSSL >= 3.0.
- SELinux: Enforcing mode is supported, but not required.

Tested Platforms (Linux)

- **OS (Intel Processors)**:
  - Ubuntu 16.04+, Debian 9+, RHEL 8+, CentOS 8+, SUSE Linux Enterprise Server 12+, Amazon Linux 2014.03+.
- **OS (ARM64 Processors)**:
  - Ubuntu (14.04, 16.04, 18.04, and 20.04), CentOS (8. 9), and Amazon Linux 2.

As of October 16, 2024, Cribl has deprecated support for the CentOS Linux 7 distro, which reached its end of life on June 30, 2024

Windows Requirements

Cribl Edge supports these versions of Windows:

Version	Workstation Type	Notes
Windows 10 and 11	Always-on network connection to the Leader.	Support for Windows workstations that do not have power management features enabled (sleep or hibernate).
Windows 10 and 11	**(Preview Feature)** Laptop or desktop with power management enabled.	Keep these things in mind when deploying Cribl Edge on Windows 10 and 11 workstations that are **not** always on: • 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. • Excessive logging has been observed when a Leader is unavailable to receive heartbeat messages from an Edge Node.
Windows Server 2016	Any	N/A
Windows Server 2019	Any	N/A
Windows Server 2022	Any	N/A

Browser Support

- Chrome, Firefox, Safari, and Microsoft Edge: the five most-recent versions.

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.

Leader/Edge Nodes Compatibility

Leaders on v.4.2.x are compatible with Edge Nodes on v.4.1.2, v.4.1.3, and later. Due to a security update, Edge Nodes running on v.4.0.4 cannot receive configurations from v.4.2.x Leaders. For details, see Leader and Edge

Nodes Compatiblity.

Disable SNI Routing for Firewalls/Web Proxies

If you have a proxy or firewall deployed between your Edge Nodes and the Leader, a new setting allows you to disable SNI routing for specific Fleets. For details, Disable SNI Routing for Firewalls/Web Proxies.

About Licensing

- **Free**: Entitles users to manage up to 100 Edge Nodes, in a single Fleet, and ingest up to 1TB/day.
- **Enterprise**: Entitles users to manage unlimited Edge Nodes, across unlimited Fleets and ingest up to its rated volume.

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, etc.

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 Controlling Metrics in Cribl Edge.

How Edge Works with AppScope

AppScope is an Open Source utility from Cribl that can send events, metrics, and/or payloads from a process or application to an AppScope Source on Edge.

Here's what to keep in mind when you plan to use Edge and AppScope together:

- AppScope is bundled with Edge and does not need to be installed.
- Running Edge with AppScope requires that Edge either run as root on a Linux host, or in a Docker container that is in privileged mode. (If this is a problem for you, see this workaround).
- To work with an AppScope host, all Edge nodes in a Fleet should be either all in Docker containers, or, all directly on Linux hosts. Plan to keep a ratio of one AppScope Source to one Fleet.

Fleet Checklist

This section compiles basic checkpoints for successfully launching a Fleet.

1. System

- 1 Leader Node (recommended: 8 vCPU/16 GB RAM).
- N Edge Nodes (depending on your environment).
- Free license, or acquire an Enterprise/Trial License from the Cribl Sales Team.

2. Configure Leader Node

- Install `git`, if not present (`yum install git`).
- Open the necessary ports.

- Download, install, and launch Cribl Edge (Linux, Windows).
- Configure as a Leader.
- Enable Start on Boot.
- Install License.

3. Configure Edge Nodes

- Enable GUI Access. Administrators will need to connect to the TCP:9420 port on each Node.
- Download, install, and launch Cribl Edge (Linux, Windows).
- Configure as a Managed Edge Node.
    - Point to the Leader address (optionally, use the configured access token).
    - Give each Edge Node an arbitrary tag like `POV`.
- Enable Start on Boot.

4. Map Edge Nodes to Fleets

- On the Leader Node, create a Fleet.
    - Name the Fleet (abitrarily) `POV`.
- On the Leader Node, confirm that Edge Nodes are connecting.
    - From the Leader Node's sidebar, select **Edge Nodes**.
- Map Nodes to the `dev` Fleet.
    - Use the Filter to select the tag you applied when configuring Nodes: `cribl.tags.includes('POV')`.

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, select **Limits**, and then select **Others**.
2. Enter the number of Edge Nodes in the **Edge Nodes count** field.

Configure Connection Processes

To configure the connection processes for your Edge Nodes:

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.

3.1.1. Planning 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.

In version 4.7 and newer, 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**. To support this higher Node limit, we've made some changes to Cribl Edge.

This guide covers:

- The specific Leader configuration we recommend for best performance.
- The changes you can expect to see in the Cribl Edge UI.
- Some areas where you might encounter slowness as we add more Node support.

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 Updates

To support Fleets that contain a large number of Nodes, we've made some UI changes. This section will highlight the key changes and limitations.

Searching for Nodes in the Explore Tab

The **Node to explore** field 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. (Note: Only the first 10,000 mappings are displayed.)

Current Limitations

We're continuing to address places where we've noticed an altered UI experience. This is what we've identified so far:

- **Logs**: The drop-down where you can select an Edge Node will display a maximum of 100 Nodes.
- **Map view**: To optimize performance, this page no longer relies on the Leader's metrics store. We've removed the **Chart** and **During** fields as a result.
- **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.

3.2. Installing Cribl Edge On Linux

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

Next, ensure that required ports are available (see Network Ports).

Then, you can install and run any of the following:

- Single Cribl Edge instance.
- Cribl Edge and Cribl Stream on the same host.
- Multiple installations of the same product, on the same host.

Single Cribl Edge Instance

These instructions explain how to run Cribl Edge locally, as a single-instance deployment on your own machine.

Un-tar the install package in a directory of choice, and rename the resulting `cribl` directory as `cribl-edge`, e.g.:

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

Set the renamed directory, (e.g., `/opt/cribl-edge/`), as your `$CRIBL_HOME` directory.

To make the `$CRIBL_HOME` env 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.

Next, navigate to `$CRIBL_HOME/bin`. Here, you can use `./cribl` commands to:

- **Set to Edge mode**: `./cribl mode-edge`
- **Start**: `./cribl start`
- **Stop**: `./cribl stop`
- **Reload**: `./cribl reload`
- **Restart**: `./cribl restart`

- **Get status**: `./cribl status`

You can change the hostname and port, by adding `-H` (address) and `p` (port options). E.g: `./cribl mode-edge -H 0.0.0.0 -p 8123`

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.

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
```

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
   ```

   Run Cribl Edge as a non-privileged user:

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

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
```

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
```

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
```

   If you are setting up Cribl Edge in single-instance mode, make sure to set `mode-edge` instead of `mode-managed-edge` .

   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
```

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`.)

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/
```

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

Cribl University's Troubleshooting Criblet on Switching from Cribl Edge to Cribl Stream demonstrates these techniques for integrating Edge with Stream. 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 Troubleshooting Criblets and Advanced Troubleshooting short courses.

3.2.1. Enabling Start On Boot

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]
```

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
```

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
```

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

```
[sudo] systemctl enable cribl-edge
```

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

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

Other available `systemctl` commands are:

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

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
```

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:

```
systemctl edit cribl-edge
```

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
```

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
```

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
```

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
```

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
```

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
```

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

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

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

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

Persisting Overrides on initd

Notes on preserving required permissions across restarts and upgrades:

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`

Important: 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.

3.2.2. Running Edge As An Unprivileged 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.

Privileges, Edge, and AppScope

When you need to run Edge as an unprivileged user, you can use the AppScope CLI to send data to Edge (instead of "driving" AppScope from the Edge UI). With this workaround, the unpriviledged Edge can still:

- Monitor status of scoped processes
- Receive data from scoped processes
- Change the configuration of scoped processes

See the AppScope CLI documentation.

3.2.3. Installing Edge On Linux Via RPM

If you need to install Edge in a tightly managed Linux environment, use 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.

RPM Download Links

Cribl Edge Version	RPM Download Link	GPG Key

4.9.1	• x64
	  ◦ SHA256
	  ◦ MD5
	• arm64
	  ◦ SHA256
	  ◦ MD5
		4.9.1 RPM Public Key

4.9.0	• x64
	  ◦ SHA256
	  ◦ MD5
	• arm64
	  ◦ SHA256
	  ◦ MD5
		4.9.0 RPM Public Key

4.8.2	• x64
	  ◦ SHA256
	  ◦ MD5
	• arm64
	  ◦ SHA256
	  ◦ MD5
		4.8.2 RPM Public Key

Cribl Edge Version	RPM Download Link	GPG Key

4.8.1	• x64
	  ◦ SHA256
	  ◦ MD5
	• arm64
	  ◦ SHA256
	  ◦ MD5
		4.8.1 RPM Public Key

4.8.0	• x64
	  ◦ SHA256
	  ◦ MD5
	• arm64
	  ◦ SHA256
	  ◦ MD5
		4.8.0 RPM Public Key

4.7.3	• x64
	  ◦ SHA256
	  ◦ MD5
	• arm64
	  ◦ SHA256
	  ◦ MD5
		4.7.3 RPM Public Key

4.7.2	• x64
	  ◦ SHA256
	  ◦ MD5
	• arm64
	  ◦ SHA256
	  ◦ MD5
		4.7.2 RPM Public Key

4.7.1	• x64
	  ◦ SHA256
	  ◦ MD5
	• arm64
		4.7.1 RPM Public Key

Cribl Edge Version	RPM Download Link	GPG Key
	◦ SHA256 ◦ MD5	
4.7.0	• x64 ◦ SHA256 ◦ MD5 • arm64 ◦ SHA256 ◦ MD5	4.7.0 RPM Public Key
4.6.1	• x64 ◦ SHA256 ◦ MD5 • arm64 ◦ SHA256 ◦ MD5	4.6.1 RPM Public Key
4.6.0	• x64 ◦ SHA256 ◦ MD5 • arm64 ◦ SHA256 ◦ MD5	4.6.0 RPM Public Key
4.5.1	• x64 ◦ SHA256 ◦ MD5 • arm64 ◦ SHA256 ◦ MD5	4.5.1 RPM Public Key
4.5	• x64	4.5.0 RPM Public Key

Cribl Edge Version	RPM Download Link	GPG Key
	◦ SHA256 ◦ MD5 • arm64 ◦ SHA256 ◦ MD5	

Verifying the RPM Package Signature

To verify the RPM package signature, enable local GPG package checking. Next, download and import the GPG key and verify its signature.

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
```

Download and Import the GPG Key

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

1. Download the GPG key: https://cdn.cribl.io/dl/CRIBL_RPM_PUBLIC_GPG.
2. Manually import the public GPG key to `rpm` using this command:

   ```
   rpm --import <URL/filepath>
   ```

   For example:

```
rpm --import https://cdn.cribl.io/dl/4.5.0/CRIBL_RPM_PUBLIC_GPG
```

3. Verify the package signature using this command:

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

   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 one of the commands, depending on the processor:

   ```
   sudo yum install https://cdn.cribl.io/dl/4.5.0/cribl-edge-4.5.0-1a628515-linux-x
   ```

   Or

   ```
   sudo yum install https://cdn.cribl.io/dl/4.5.0/cribl-edge-4.5.0-1a628515-linux-a
   ```

   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_MASTER_URL=tls://<authToken>@leader:4200`
3. Enable the service:

- `systemctl enable cribl-edge` to enable the service on boot.
- `systemctl start cribl-edge` to start it manually.

Upgrading Edge via RPM

See Upgrading Edge via RPM.

Uninstalling Edge via RPM

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

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

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.

3.2.4. Anti-Virus Exceptions

If you are running anti-virus software on a 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.

3.3. Installing Cribl Edge On Windows

You can install Cribl Edge in a supported Windows environment.

Supported Windows Versions

Cribl Edge supports these versions of Windows:

Version	Workstation Type	Notes
Windows 10 and 11	Always-on network connection to the Leader.	Support for Windows workstations that do not have power management features enabled (sleep or hibernate).
Windows 10 and 11	**(Preview Feature)** Laptop or desktop with power management enabled.	Keep these things in mind when deploying Cribl Edge on Windows 10 and 11 workstations that are **not** always on: • 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. • Excessive logging has been observed when a Leader is unavailable to receive heartbeat messages from an Edge Node.
Windows Server 2016	Any	N/A
Windows Server 2019	Any	N/A
Windows Server 2022	Any	N/A

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 Network 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 concatenate and copy/paste the bootstrap script in your command prompt to add Windows Nodes.

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.

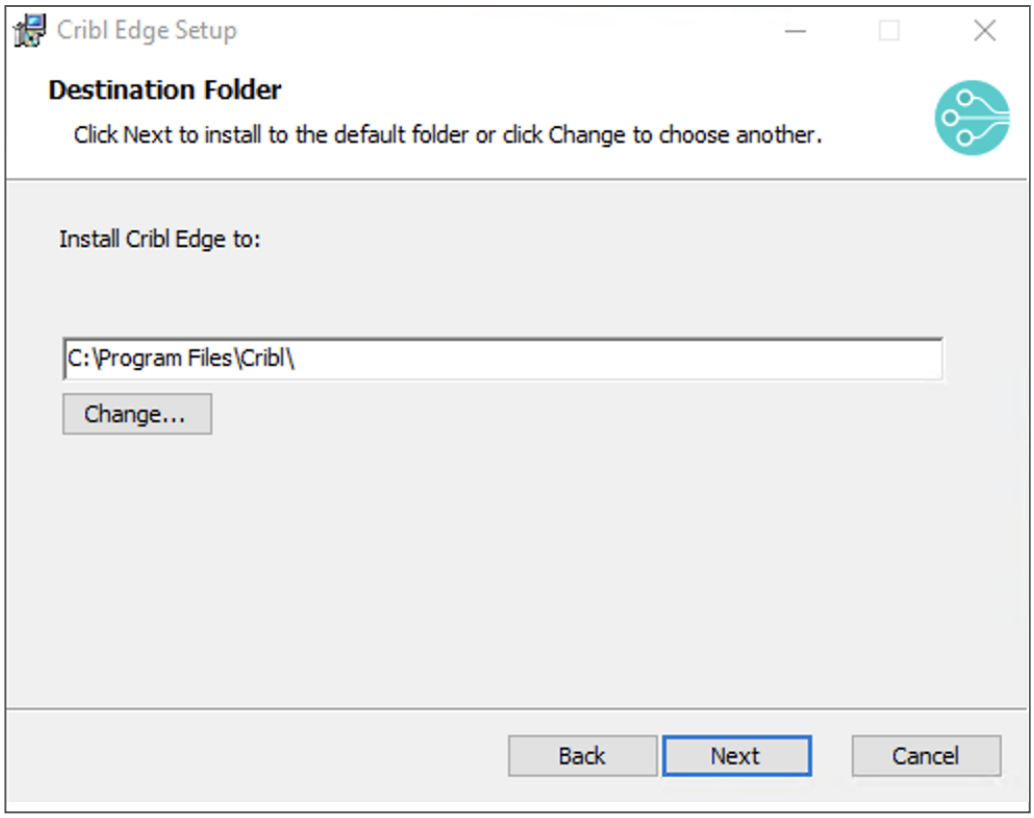
Either method installs Cribl Edge as a Windows service. This enables Cribl Edge to automatically restart whenever the Windows Server reboots.

If you are on Cribl.Cloud, the Enabling TLS After Installation section will walk you through creating an `instance.yml` file upon initial Cribl Edge/Windows installation, and then copying it to the same location for each subsequent install.

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

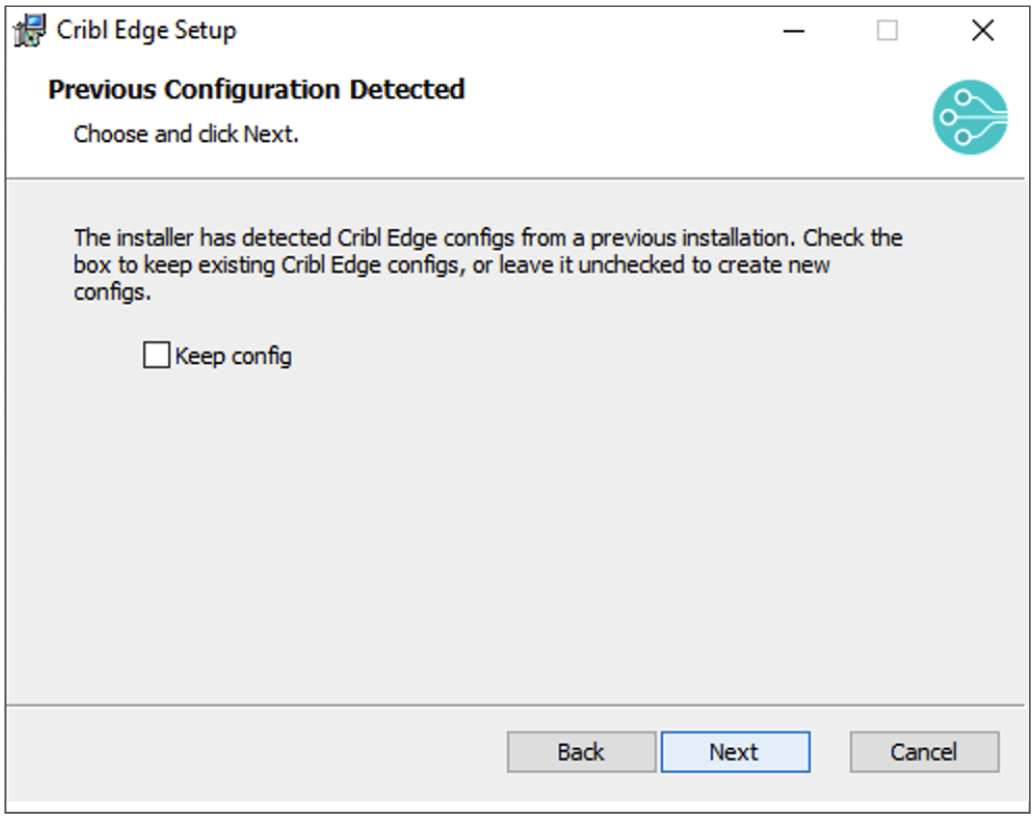
Use the Wizard

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

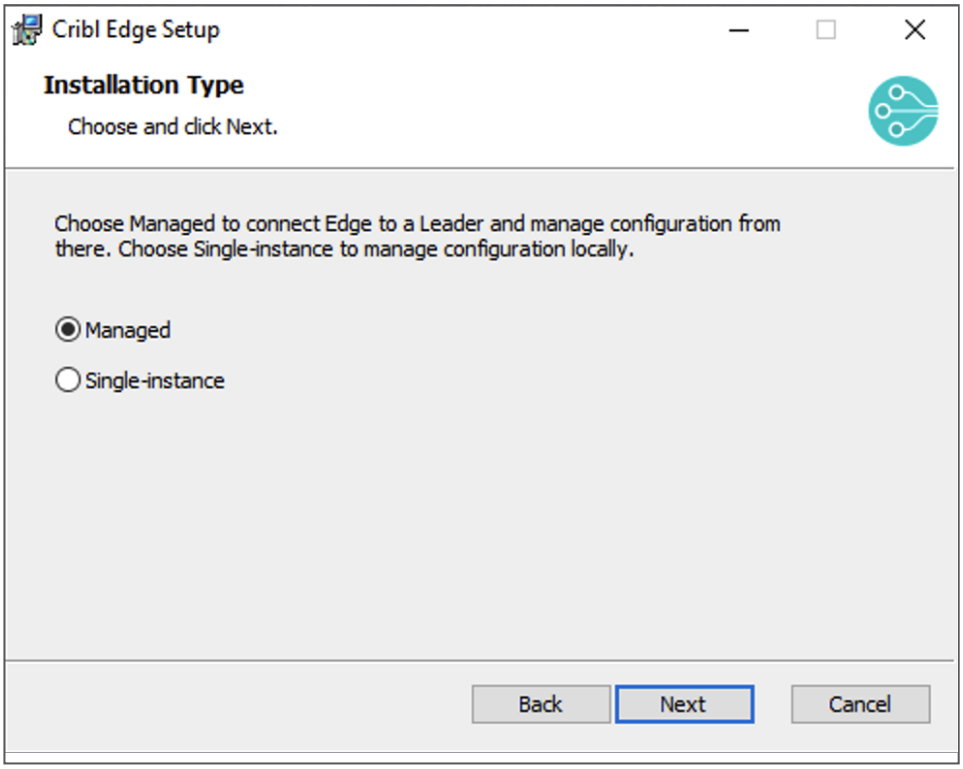
4. If the installer detects a config file from a previous Cribl Edge installation, the modal will ask you if you want to keep your current config or create a new config file for the installation. Select **Keep config** to persist your previous configurations stored in the `instance.yml` file.



Keep Config

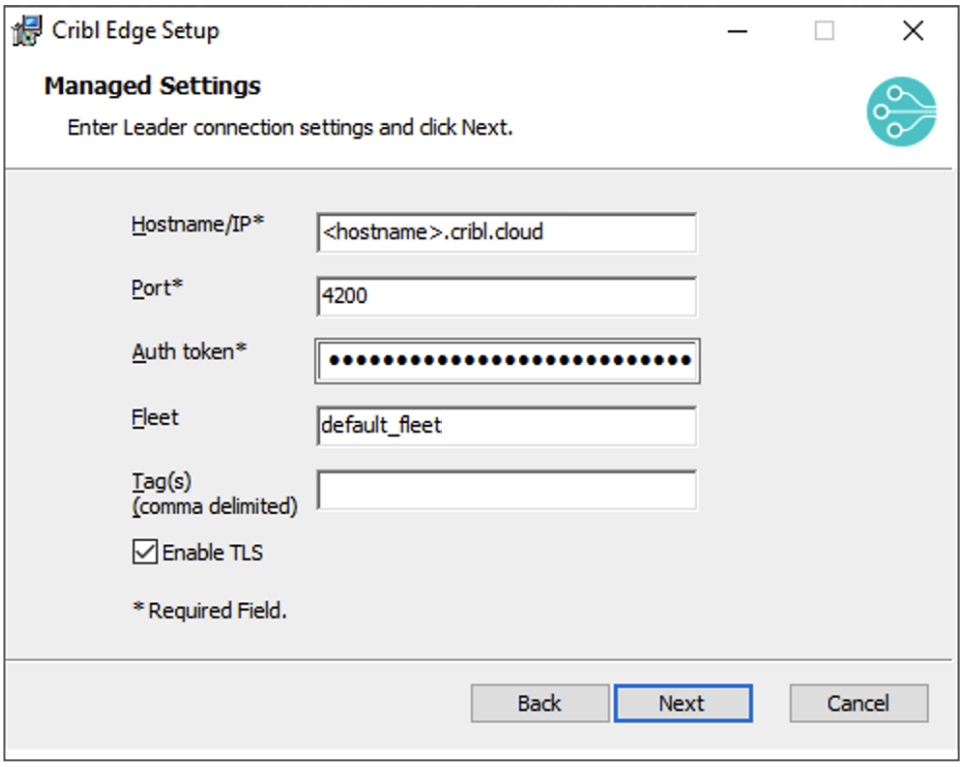
5. Select either **Managed** or **Single-instance** as the installation type. See Types of Deployment for information on these options.
   If you chose **Keep config** in the previous window, the installer will skip the installation type selection.

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



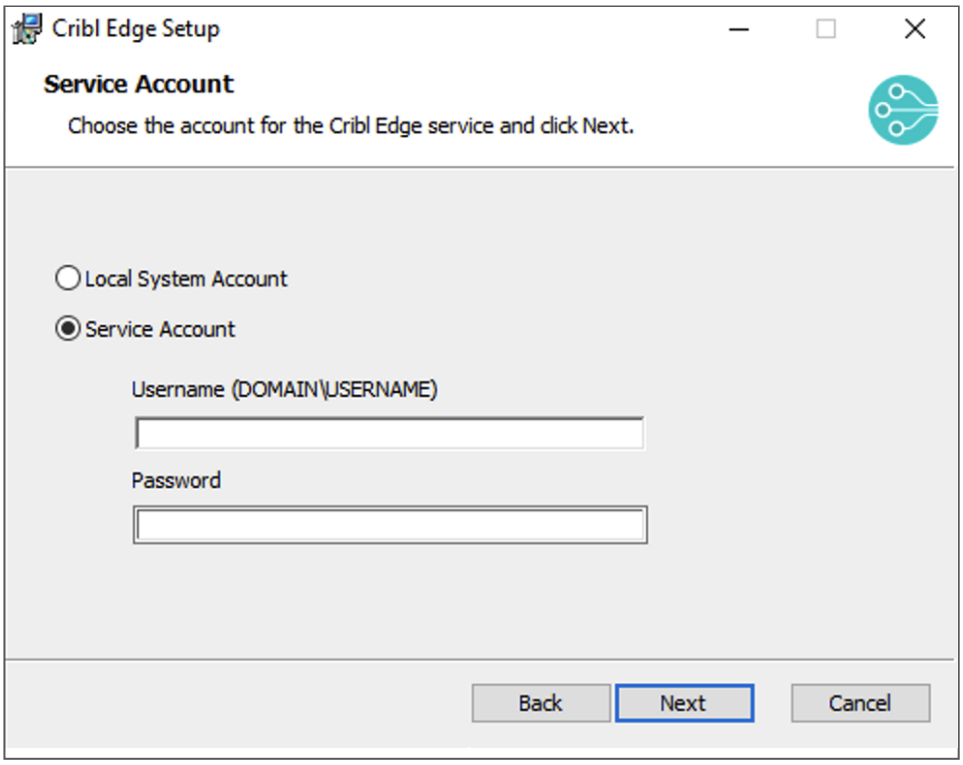
Select installation type

6. If you select **Managed** mode, enter the Leader hostname or IP address, port, auth token, and optionally the Fleet and any tags.

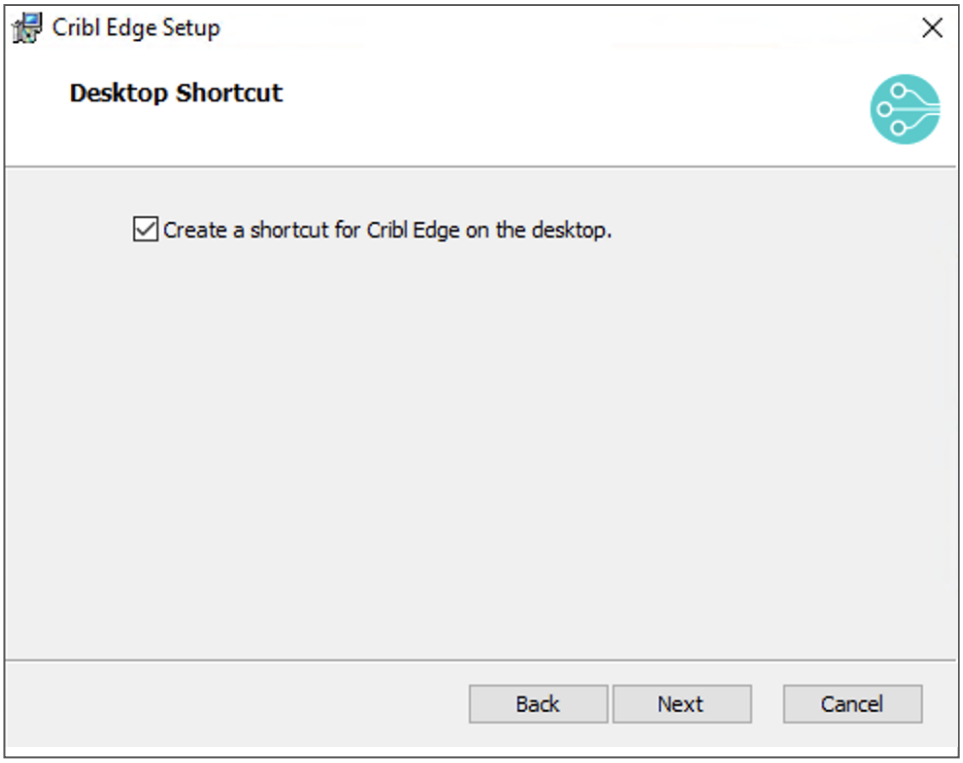


Connection Details

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**.

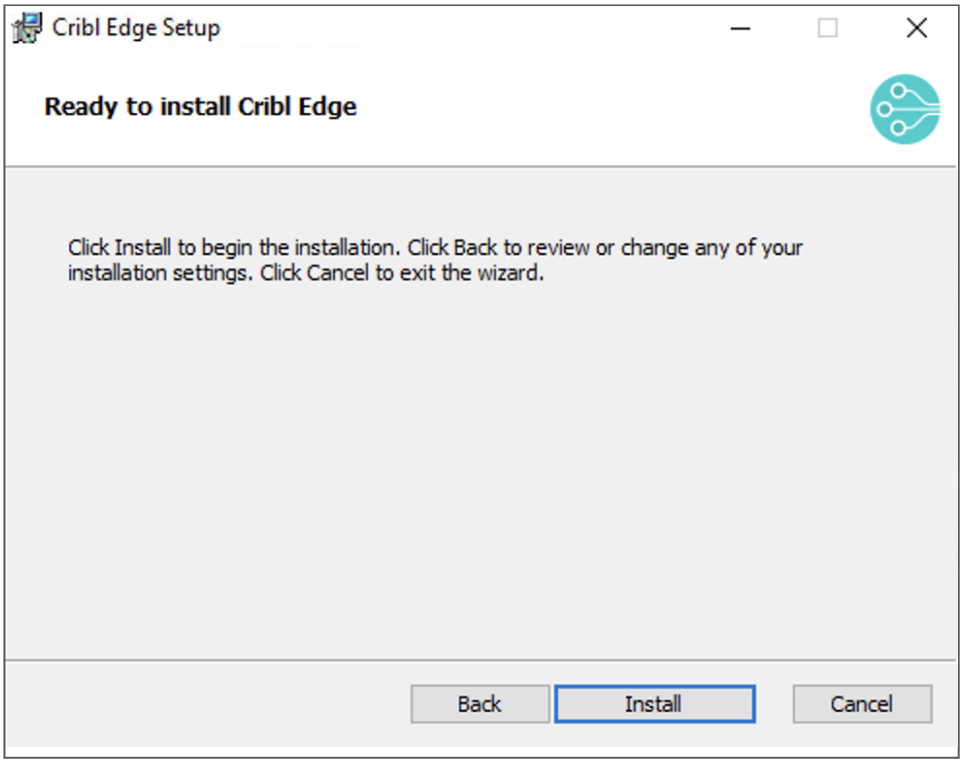


8. Optionally **Create a desktop shortcut**.



Desktop shortcut option

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



Installation summary

10. When installation is complete, double-click the Cribl Edge icon on your Desktop or File Explorer. This will launch Cribl Edge's login page in your default browser. Or go to `http://localhost:9420` and log in with default credentials (`admin:admin`).

As of Cribl Edge v.4.0, the modified data (including the `instance.yml`) now goes to `C:\ProgramData\Cribl` instead of `C:\Program Files\Cribl`.

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 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.

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

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

For Cribl Edge v.4.1.0 or later, use the flag `KEEPDATA=1` to persist data between installs. `KEEPDATA` will not overwrite the existing configurations, state, logs, etc.

```
msiexec /i cribl-<version>-<build>.msi /qn KEEPDATA=1
```

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

If you are installing Cribl Edge into a Docker container on Windows, you must include the `username`. For example:

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

Enable TLS After Installation

Some on-prem deployments don't require secure (TLS) communication between the Leader and Edge Node. In these cases, the Managed Node will complete its configuration, followed by removing the `admin/admin` password. There will be no reason to locally log in, as you can manage the Edge Node via the Leader UI.

For sites using secure TLS connections to the Leader, including Cribl.Cloud, you must configure the local Edge Node to enable TLS. To do this:

1. Log into your local instance (at `http://localhost:9420`) with the default credentials (`admin/admin`).
2. Enable TLS locally. For details, see Secure Leader/Edge Nodes Communication.
3. When prompted, restart your Cribl Edge instance.
4. Locate the `instance.yml` file in:
    - Cribl Edge v.4.0.X or earlier: `Program Files\cribl\local\_system`.

- Cribl Edge v. 4.1 or later: `C:\ProgramData\Cribl`.

To avoid this post-installation update for subsequent Windows installations to other servers, you can copy the `instance.yml` from this server. After the installation is complete, paste the `instance.yml` file into subsequent servers' `\Program Files\cribl\local\_system` folder.

To connect each new Cribl Edge instance to this config file, you'll need to either restart the Windows Server, or simply run the following commands as a Windows Administrator:

```
net stop cribl

net start cribl
```

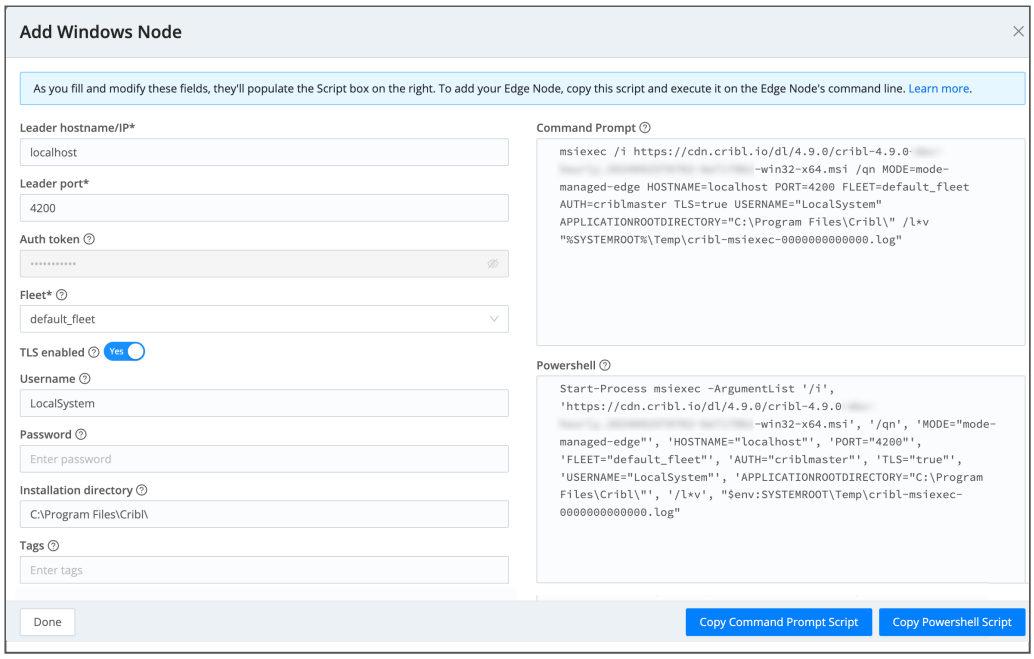
Add or Update a Windows Node

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 Windows Command Prompt or Powershell.
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 (`/l*v "%SYSTEMROOT%\Temp\cribl-msiexec-0000000000000.log"`). You can replace the zeros with an epoch timestamp (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 `/l*v "%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 Linux epoch time in milliseconds as the timestamp.

Troubleshooting and Logging

Here are some helpful tips:

Setting Logging Options

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

Changing the Installation Directory

If you want to change the installation directory, use the `msiexec` command `APPLICATIONROOTDIRECTORY`. For example:

```
msiexec.exe /i Cribl-Edge.msi APPLICATIONROOTDIRECTORY="C:\test\" /L*V "C:\Log\Cribl
```

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

Frequently Asked Questions

Here are some common questions about Cribl Edge on Windows, and the answers!

Q: What is the `nssm.exe` process?

A: `nssm.exe` is a wrapper for the `cribl.exe` binary. It gives Windows Service Manager the ability to manage an executable program without having to provide a number of required binary APIs on the executable. So, instead of the `cribl.exe` binary needing to provide the APIs, `nssm.exe` acts as an intermediary, providing the necessary APIs itself and subsequently spawning `cribl.exe` as a child process.

Q: Why is `nssm.exe` important?

A: The `nssm.exe` process ensures that Cribl Edge is stable and reliable on Windows, both by booting up Cribl Edge and restarting it for you. With the `nssm.exe` process running, Cribl Edge remains operational even if it closes unexpectedly. While some antivirus software might flag `nssm.exe` due to its monitoring and restart functionality, `nssm.exe` is a legitimate and essential component of Cribl Edge.

High CPU utilization may appear shortly after launching Cribl Edge but is not necessarily related to `nssm.exe`. We recommend waiting for Cribl Edge to fully boot and stabilize before checking CPU usage.

3.3.1. System Proxy Configuration 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 few ways to do it, including using Windows system settings, Group Policy Objects, and environment variables.

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.

Environment Variables

Another way to enable proxy is to set the `HTTP_PROXY` and `HTTPS_PROXY` environment variables on the Cribl Edge Windows Service before starting Cribl Edge.

Configure the 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.

```
$environment = [string[]]@("HTTP_PROXY=http://proxy:1234","HTTPS_PROXY=http://proxy:1
Set-ItemProperty HKLM:SYSTEM\CurrentControlSet\Services\Cribl -Name Environment -Valu
```

In the above example, note that when you set an `HTTPS_PROXY` environment variable, the referenced URL should generally be in `http` format.

Restarts and Case Conflicts

You must restart Cribl Edge on the affected Nodes if the application is running when you first configure variables or make changes to them.

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.

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. Specify the user name and password in the proxy URL. For example:

```
$environment = [string[]]@("HTTP_PROXY=http://username:password@proxy:1234","HTTPS_PI
Set-ItemProperty HKLM:SYSTEM\CurrentControlSet\Services\Cribl -Name Environment -Valu
```

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>"
```

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.

3.4. Running In A Docker Container

As a best practice, we recommend using the UI to concatenate and copy/paste the bootstrap script for adding a Docker node.

Or, for a single-instance deployment, start the container with the Docker command below:

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

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

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

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

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

The `-v /var/run/appscope:/var/run/appscope \` line exposes the Unix domain socket for the AppScope Source so that it's accessible to the `scope` application on the host, and in other containers that mount that same directory. See Running AppScope With Cribl Edge in a Docker Container in the AppScope documentation.

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

Running 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_MASTER_URL=tcp://INSERT_TOKEN@leaderhere:4200?group=fleet_name
 -e "CRIBL_EDGE=1"
 -e CRIBL_VOLUME_DIR=/tmp/
 -v /var/run/appscope:/var/run/appscope \
 -v /var/run/docker.sock:/var/run/docker.sock \
 -v /:/hostfs:ro \
 -p 9420:9420
 --restart unless-stopped \
 --name cribl-edge \
 cribl/cribl:latest
```

Overriding Default Ports

Default ports can be overridden 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.

Updating the Docker Image and the Packaged OS

For details, see the following topics:

- Updating the Docker Image
- Updating the Packaged OS

Troubleshooting

When you deploy containers via Kubernetes in `managed-edge` mode, this Node will – after applying the first config bundle from the Leader – listen on the Fleet's configured **API Server Settings** > **Host**.

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.

Cribl University offers a course titled How to Install & Configure Edge Using Docker that illustrates the installation steps. 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.

3.5. Deploying Via Kubernetes

This page outlines how to 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 AppScope metadata) will tolerate invalid TLS certificates (i.e., expired, self-signed, etc.) 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`.

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.

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
```

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
```

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 update your Helm charts with appropriate security context configurations.

To add a Security Context Section to the Helm Chart:

1. Open the YAML file for your deployment within the Helm chart.
2. Navigate to the `spec.template.spec.containers` section. This is where container configurations are defined.
3. Inside your container definition, add a `securityContext` section (see below), customizing 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
```

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']
```

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']
```

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']
```

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']
```

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
```

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.

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 AppScope 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.

3.6. 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, Collectors and Collector-based Sources can continue ingesting data without interruption.

For license tiers that support configuring backup Leaders, see Cribl Pricing.

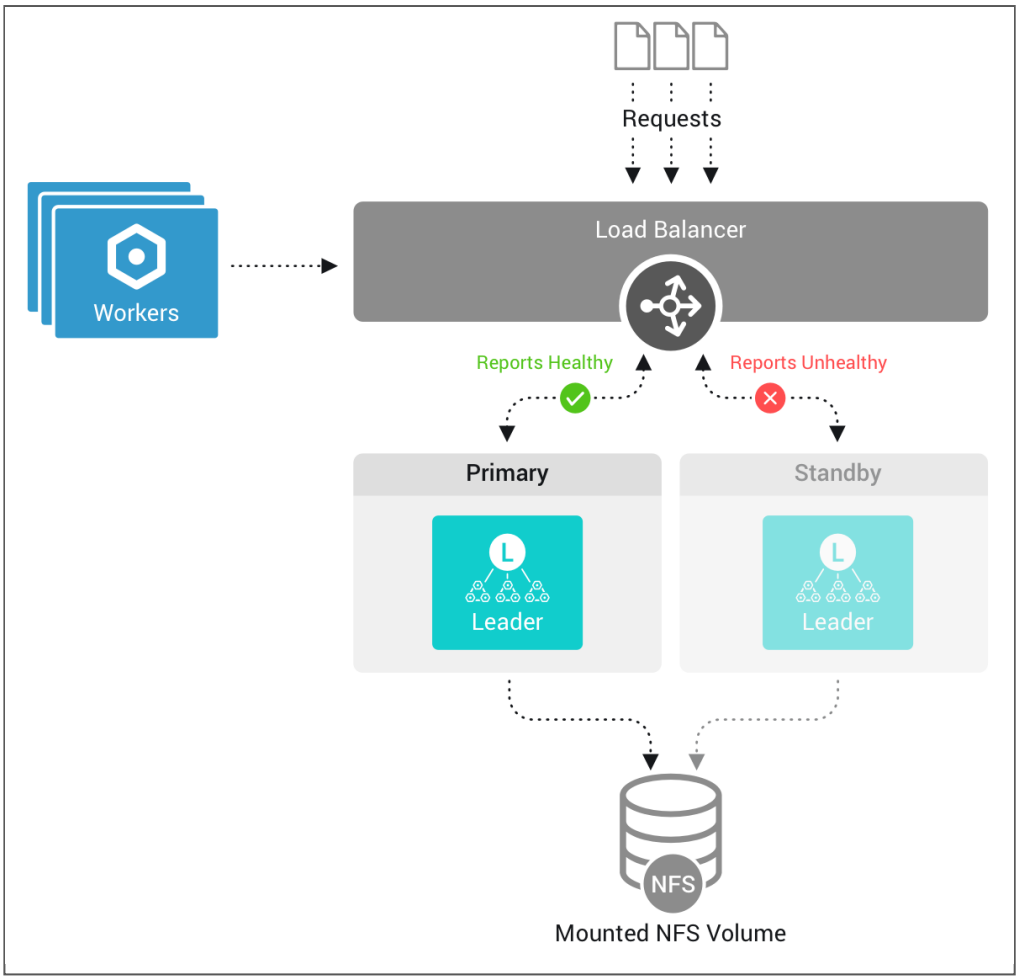
How It Works

There is only ever one active Leader Node at one time. A standby Leader will become active only in the event of failover. In the configuration for all Leaders, you must specify the same **failover volume** – a shared Network File System (NFS) volume.

If the primary Leader Node goes down:

- Cribl Edge will recover by switching to a standby Leader.
- The new Leader will have the same configs, state, and metrics as the previous Leader Node.
- The Edge Nodes connect to the new Leader.

Prior to Cribl Edge 4.7, it was possible for the primary and standby Leaders to be configured differently, potentially causing issues, especially around authentication and authorization.



Leader High Availability/Failover Design

Required Configuration

Before adding a standby Leader, ensure that you have the configuration outlined in this section.

Auth Tokens

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 and match these values at each Leader's **Settings** > **Global** > **Distributed Settings** > **Leader Settings** > **Auth token**.
- Or, from the filesystem, check and match all Leaders' instance.yml > `master` section > `authToken` values.

See How to Secure the Auth Token for the Leader Node for information on changing your auth token and allowed characters.

NFS

- 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.

You can validate the NFS latency using a tool like `ioping`. Navigate to the NFS mount, and enter the following command:

```
ioping .
```

For details on this particular option, see the ioping docs.

NFS Mount Options

The Leader Node will access large numbers of files whenever you use the UI or deploy configurations to Cribl Edge 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.

Load Balancers

- Configure all Leaders behind a load balancer.
- Expose ports `9000` and `4200` via the load balancer.
- Load balancers must support health checks over HTTP/HTTPS via the `/api/v1/health` endpoint.

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.

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?

Recommended Configuration

Use the latest NFS client across all Leaders. 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.

Set the primary Leader's **Resiliency** drop-down to `Failover`.

Configure Additional Leader Nodes

You can configure additional 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`.

How Cribl Edge Manages Leader Settings

When you first configure a Leader for failover, 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 `leader.yml` file on the failover volume will take precedence over what is stored in the local `instance.yml` file.

The `leader.yml` file replicates most of the content of the local `instance.yml`, but leaves out the failover configuration.

While running in failover mode, when you change **Settings** > **Distributed Settings** via the UI, Cribl Edge applies those changes to `leader.yml` in the failover directory.

Use the UI

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 to support Leader failover. 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** from its default `5s`. This setting determines how often the primary Leader tries to refresh its hold on the Lease file.
5. Optionally, adjust the **Missed refresh limit** from its default `3`. This setting determines how many Lease refresh periods elapse before standby Nodes attempt to promote themselves to primary.
6. Select **Save** to restart.

In Cribl Edge 4.0.3 and later, when you save the **Resiliency**: `Failover` setting, further **Distributed Settings** changes via the UI will be locked on 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, as covered in the very next section.

Use the YAML Config File

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.

$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
```

Note that `instance.yml` configs are local, not on the shared NFS volume.

Use the Command Line

You can configure another Leader Node using a CLI command of this form:

```
./cribl mode-master -r failover -v /tmp/shared
```

For all options, see the CLI Reference.

Use Environment Variables

You can configure additional Leader Nodes via the following environment variables:

- `CRIBL_DIST_MASTER_RESILIENCY=failover`: Sets the Leader's `Resiliency` to `Failover` mode.
- `CRIBL_DIST_MASTER_FAILOVER_VOLUME=<shared_dir>`: Sets the location of the NFS directory to support Leader failover – for example, `/mnt/cribl`.
- `CRIBL_DIST_MASTER_FAILOVER_MISSED_HB_LIMIT`: Determines how many Lease refresh periods elapse before standby Nodes attempt to promote themselves to primary. Cribl recommends setting this to `3`.
- `CRIBL_DIST_MASTER_FAILOVER_PERIOD`: Determines how often the primary Leader refreshes its hold on the Lease file. Cribl recommends setting this to `5s`.

For further variables, see Environment Variables.

Monitor the Leader Nodes

To view the status of your Leader Nodes, select **Monitoring** > **System** > **Leaders**.

Monitoring Leader Nodes

Upgrade

When upgrading:

1. Stop all Leaders.
2. Upgrade the primary Leader for your Stream or Edge deployment.
3. Upgrade standby Leaders.
4. Start each Leader again, one by one.
5. Upgrade each Edge Node, respectively.

Disable a Standby Leader

Cribl recommends that you maintain a standby Leader to ensure continuity in your on-premises distributed environment. Should you decide to disable it, contact support to assist you.

3.7. Ports

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-premises).	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).	In
`4200`	HTTP/S	Software upgrade (via path, not CDN).	In

Edge Nodes

The following ports are used by Edge Nodes.

Default Port	Protocol	Purpose	Direction
`9420`	TCP	Cribl Edge UI.	In
`4200`	HTTP/S	Config bundle downloads.	Out
`4200`	TCP	Heartbeat/Metrics/Leader requests/notifications to clients (for example: live captures, teleporting, status updates, config bundle notifications, and so on).	Out
`443`	TCP	(In Cribl.Cloud hybrid deployments) Communication with the Leader and with `https://cdn.cribl.io`.	Out

Other Ports

Common Ports

Default Port	Protocol	Purpose	Direction
53	UDP	DNS lookups.	Out
389	TCP	LDAP Auth (non-TLS).	Out
443	TCP	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.	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

Default Port	Protocol	Purpose	Direction
9090	TCP	Collection/discovery from Prometheus Scraper.	Out
9092	TCP	Collection from Confluent Cloud or Kafka, used when no port is provided.	Out
9092	TCP	Output to Confluent Cloud or Kafka, used when no port is provided.	Out
9093	TCP	Output to Azure Event Hubs.	Out
9997	TCP	Splunk TCP Source.	Out
10070	TCP	TCP JSON data.	In
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
20000 – 20010	TCP	Additional Sources in Cribl.Cloud.	In

No other ports can be opened for Cribl-managed Worker Groups in Cribl.Cloud.

4. Fleet Management

Getting started with managing Fleets of Edge Nodes in a distributed deployment

---

In a distributed environment, Edge Nodes can be managed centrally by a single Leader Node, which 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.

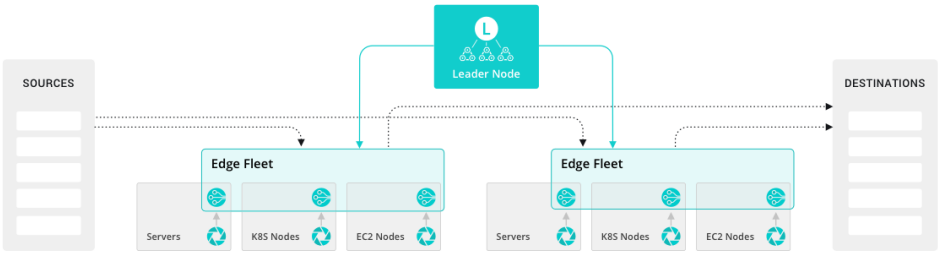
Worker Process – a Linux process within a Single Instance, or within Edge Nodes, that handles data inputs, processing, and output. The process count for Edge Nodes is constrained to 1.

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.

Installing on Linux

See Installing on Linux.

Installing on Windows

See Installing on Windows.

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.

4.1. Fleets

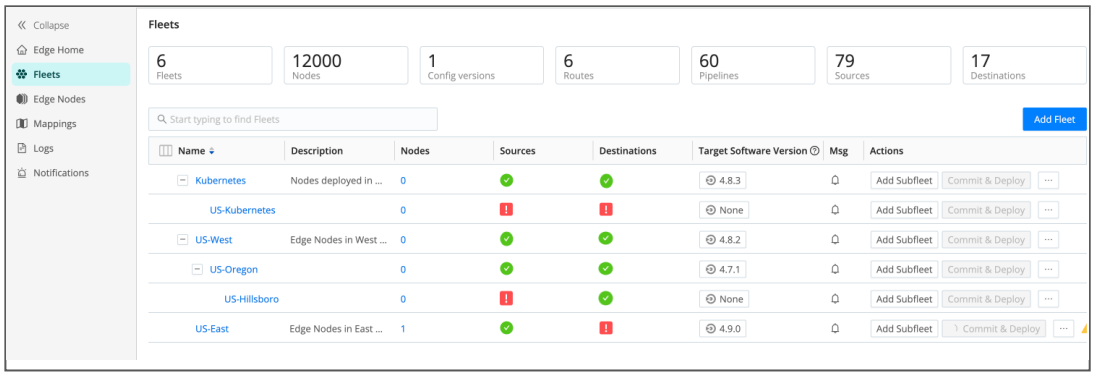
4.1.1. Explore Cribl Edge Fleets

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.

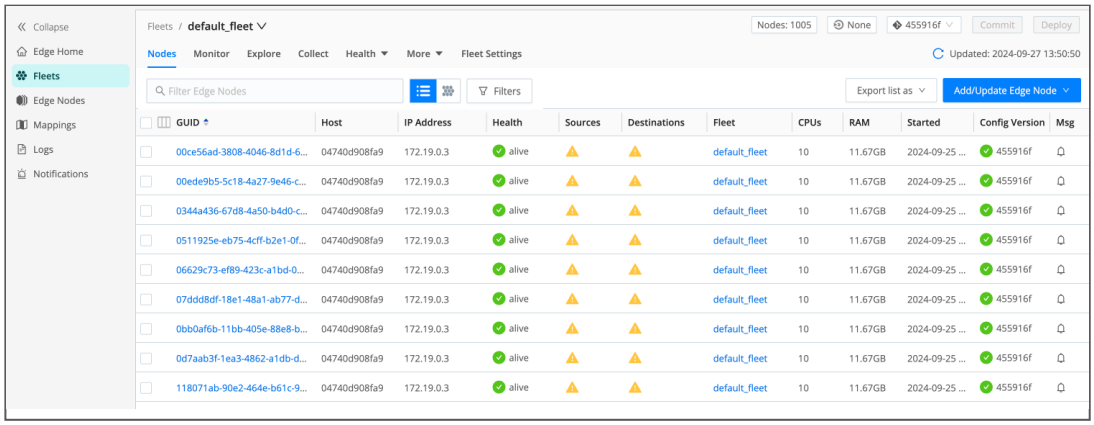


Manage Fleets

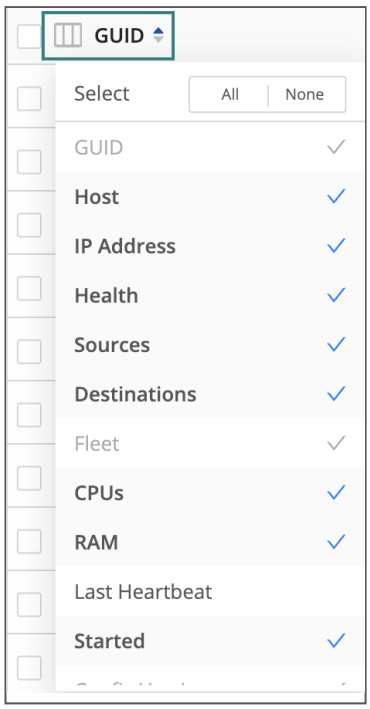
To explore more information for each of your configured Fleets, select a Fleet **Name** to open it.

View Edge Nodes in a Fleet

When you open a Fleet, you'll land on the **Nodes** page, in **List View**.



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.

Field	Description
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.

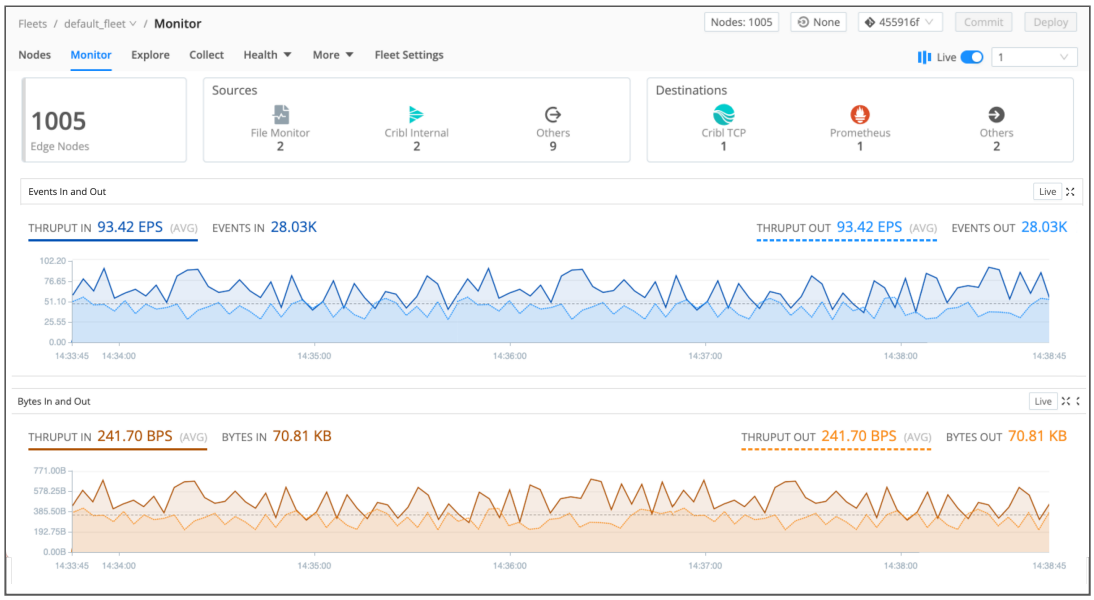
Understand the Health Status Icons

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.

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 Upgrading Edge Nodes via the UI.
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.)
Bytes In	The uncompressed amount of data in the last 5 minutes of data collected. You can change the display's granularity from the default `last 5 min`, selecting from 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.)
Sources	List of configured Sources.
Destinations	List of configured Destinations.

Ways to View Nodes

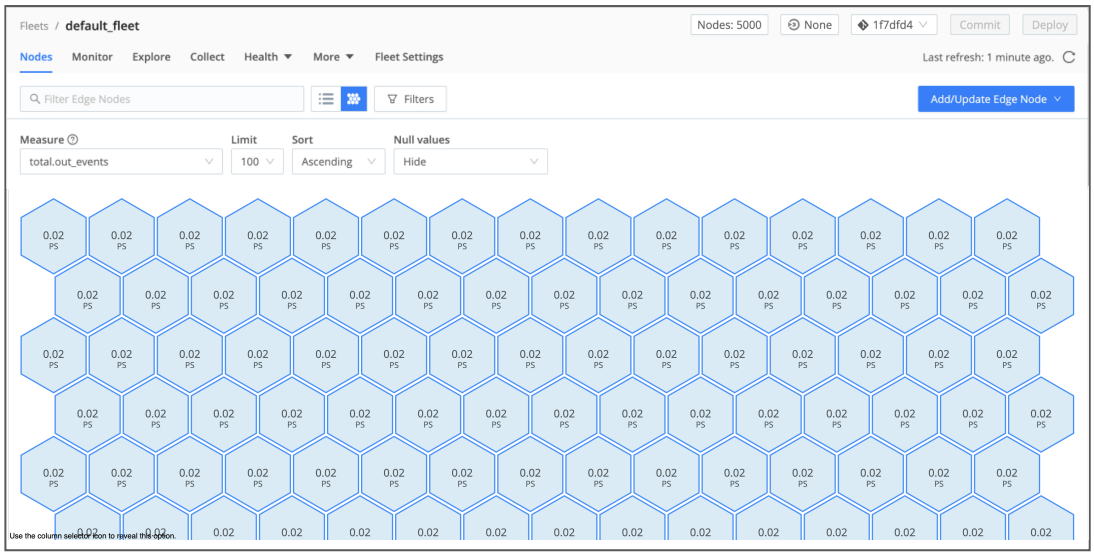
To switch between two Edge Node views, select the List View icon or the Map View icon.

List View: This is a list of all the Nodes and their statuses in the Fleet.

Map View: Here, a query builder allows you to display metrics from the Edge Nodes in the Fleet. You can select from different aggregations in the **Chart** field, different metrics in the **Measure** field, and the time window in the **During** field.

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 Controlling 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.



View Edge Node

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.

4.1.2. Fleets Hierarchy And Design

Cribl Edge enables you to flexibly group Edge Nodes into logical Fleets and Subfleets, organized as you like. However, Fleet design is extremely important for the health of the Leader. In this guide, we address "better practices" for designing your Fleets and Subfleets.

Leader Health

Below are some guidelines for designing your Fleets to optimize your Leader's health:

- 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 a larger number of Edge nodes, change the default metrics controls to `minimal`.

Fleet Design Better Practices for Enterprises

Consider the following "better practices" for designing Fleets for enterprises:

- Do not use `default_fleet`. The `default_fleet` is the fallback for mapping errors. If you purchased an Enterprise License, disable all the Sources in the `default_fleet` to prevent accidental collections. This guideline doesn't apply to the free license, where the only option is `default_fleet`.
- Design your Fleet hierarchy before deploying your Edge Nodes. For details, see Designing a Fleet Hierarchy below.
- Use the bootstrap script in the Add/Update Edge Node modal to deploy Edge Nodes to a specific Fleet. Depending on your Fleet design, we recommend adding tags to the bootstrap script to uniquely identify the Fleet. You can use these tags for automatic mappings.
- 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`. If you disable all the Sources in the `default_fleet`, there will be no impact on data consumption.
- Monitor your `default_fleet` for any unintentional/rogue Edge Nodes.

Designing Your Fleet Hierarchy

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` > `Application`. In this case, the Edge Nodes at the `Application` level can inherit and share configurations from the `Org` 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.

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, etc.): Another good way to organize your Fleets can be based on server functions.
- OS-based: If you are running an environment with both Linux and Windows machines, consider separating them into different Fleets per OS.

The recommendations here:

- To avoid confusion, try to keep Fleets to two to three levels. We'll discuss it more under Inheritance Considerations.

To learn more about Fleets, see Fleets and check out Configuring a Fleet for configuration details.

Naming your Fleets

What's in a name, you ask? Well, a whole lot when it comes to Fleets. 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 four-level Fleet in an intuitive way:

- BZT
    - BZT-NewYork
        - BZT-NY-Linux
            - BZT-NY-Linux-Apache
            - BZT-NY-Linux-Gizmotrax

        - BZT-NY-Windows
            - BZT-NY-Win-IIS
    - BZT-LA
        - BZT-LA-K8s
        - etc.

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.

Inheritance Considerations

As suggested above in Designing Your Fleet Hierarchy, plan and design the Fleets' inheritance in advance. 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. Consider:

- Applying certificates and other company-wide settings at this level.
- Avoid OS-specific or site-specific configurations at this level.

Subfleet 1 (Parents)

Per-site configurations. Consider:

- Configure site-specific Sources/Destinations at this level.
- Specify any site-specific configurations, like common Destinations.

Subfleet 2 (Children)

Configurations used by groups of similar systems. Consider:

- 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.

Subfleet 3 (Grandchildren)

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

You can add up to four levels of Subfleets underneath the parent Fleet, and each Subfleet will inherit the configurations of every parent Subfleet before it.

If you make a change to an inherited config (such as Sources/Destinations) in a Subfleet, the Subfleet will no longer get updates made to the parent Fleet for that inherited config. Other configs are still inherited from that parent.

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.

Finally, note that Subfleets don't inherit Packs and Pipelines from a parent Fleet at this time. You must recreate or import them into child Fleets.

4.1.3. Creating And Managing 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.

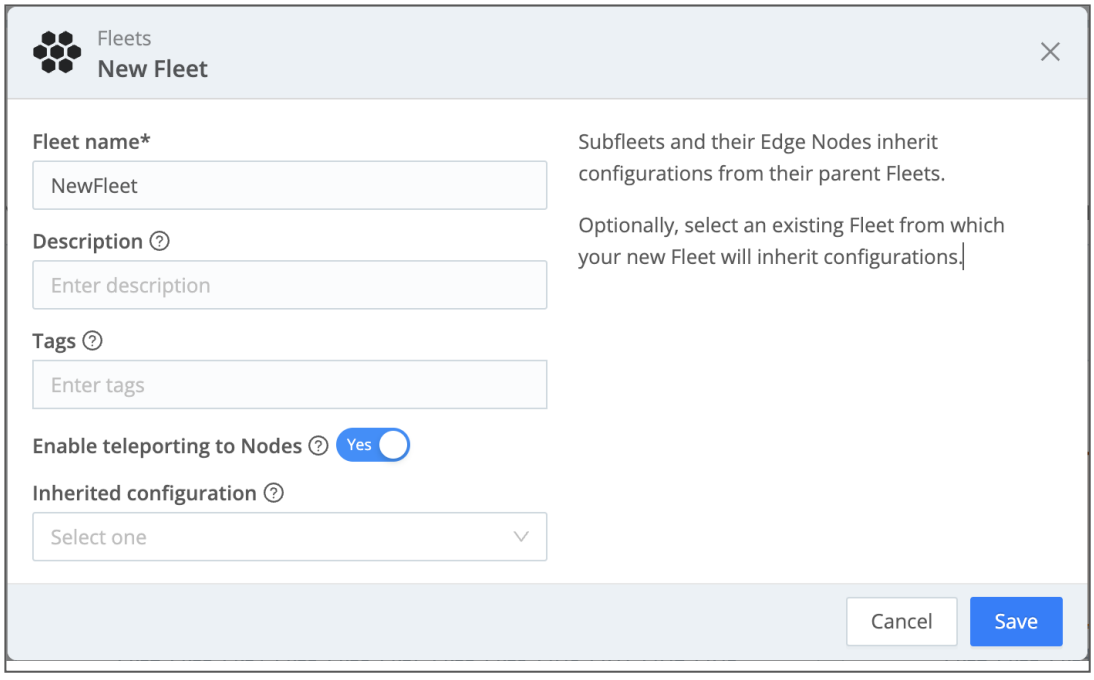
In Cribl Edge v.4.2.x and later, you can assign Fleet-level permissions to users. For details, see Members and Permissions.

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**.

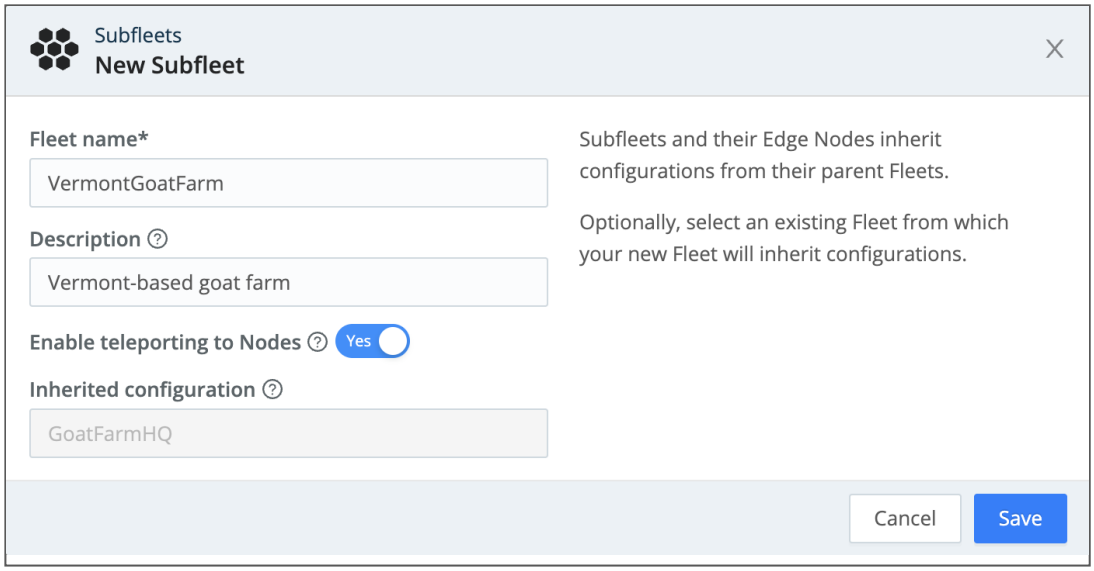
Add a Subfleet

You can create up to four levels of Subfleets nested below a parent Fleet, and choose whether the Subfleet will inherit configurations from its parent Fleet.

To add a Subfleet:

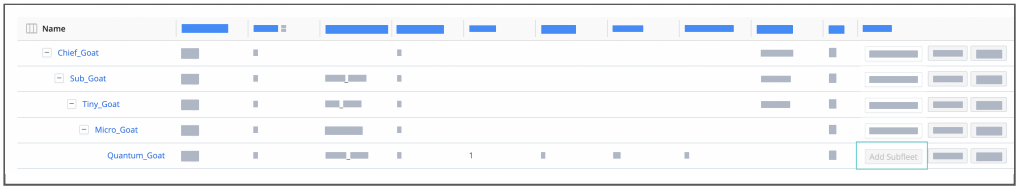
1. In the row of your desired Fleet, click **Add Subfleet** in the **Actions** column.
2. Enter a descriptive **Fleet name** and optional **Description** and Tags.
3. Select an existing Fleet from the **Inherited configuration** drop-down. The Subfleet will inherit its configuration from the selected Fleet.
4. Click **Save**.

You can also add a new Subfleet from a parent Fleet by clicking the **Add Subfleet** button in the **Actions** column.



Add a Subfleet

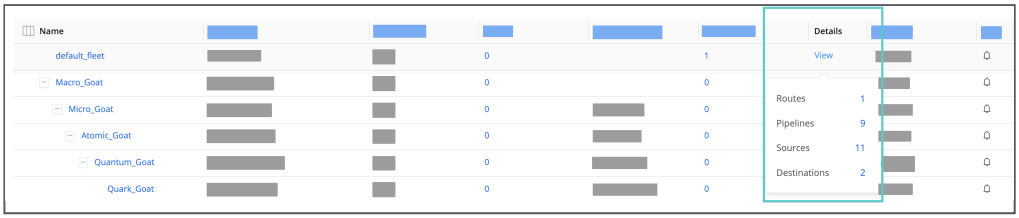
Similarly, you can create a Subfleet in a selected Subfleet. You can add up to four levels of Subfleets underneath the parent Fleet, and each Subfleet will inherit the configurations of every parent Subfleet before it.



Subfleet maximum depth

View Subfleet Details and Options

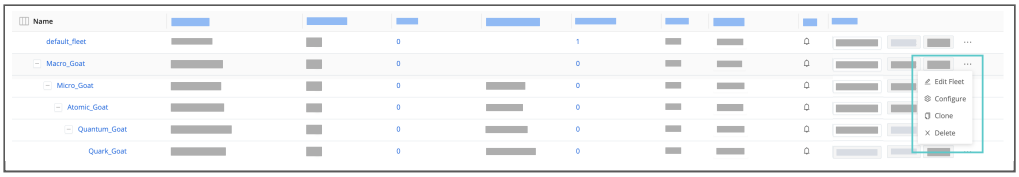
To display the number of Routes, Pipelines, Sources, and Destinations in a Fleet, click **View** in the **Details** column of a Fleet or Subfleet.



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, **Clone**, and **Delete** options, click the **Options** (⋯) menu.

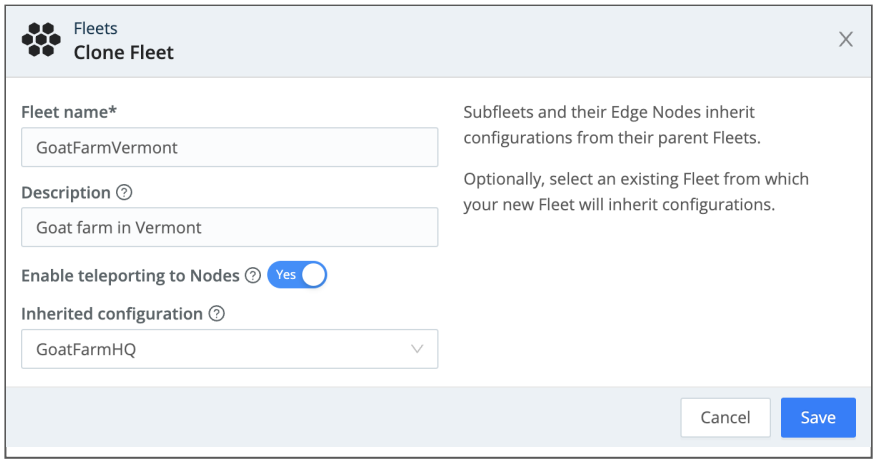


Edit, Configure, Clone, and Delete Fleets and Subfleets

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 on certain plan/license tiers. For details, see Pricing.

Subfleet Inheritance Configurations

You can organize your Fleets and Subfleets into a hierarchy of configuration layers from the top level down. Subfleets will inherit all the configurations from each parent Fleet that comes before it. At the Fleet level, this might include grouping Edge Nodes with basic configurations like common logging locations, metrics, as well as common Sources and Destinations. At the Subfleet level, you can group Edge Nodes to pick up configurations specific to the applications and services that are running on the Nodes.

There are several strategies you can use to build Fleet hierarchy:

- Organizationally.
- Geographically (if your organization collects different data categories in different regions).
- Data center–based.
- OS-based.

Layering the configurations allows 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 > Application`. In this case, the Edge Nodes at the `Application` level can inherit and share configurations from the `Org` 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. For guidance, see our better practices doc: Fleets Hierarchy and Design.

Access Inherited Packs, Lookups, and Samples

Fleets with inherited configurations can also inherit Packs, Lookups, and Samples. Here's how to access them:

- Access inherited Packs in **More** > **Data Routes**, then open the **Pipeline** drop-down in a Route. Inherited Packs don't appear in the Packs list.
- You can teleport into a Node and view Lookups and Samples in their respective lists when you deploy configuration changes to the Edge Nodes.

Save and Manage Certificates

Configure certificates at the Fleet level to securely connect your Cribl Edge to Sources and Destinations. For details, see Setting Authentication on Sources/Destinations.

Subfleets automatically inherit these certificates from their Parent Fleet, streamlining your deployment process. These certificates are then distributed to individual Edge Nodes. Certificates are stored locally on each Edge Node in the `local/edge/auth/certs/` directory.

To view and manage certificates, navigate to **Fleet Settings** > **Security** > **Certificates**.

All certificates from both the parent Fleet and Subfleet are visible in the Subfleet's certificate list.

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.

4.1.4. Configuring Fleets And Deploying 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**.

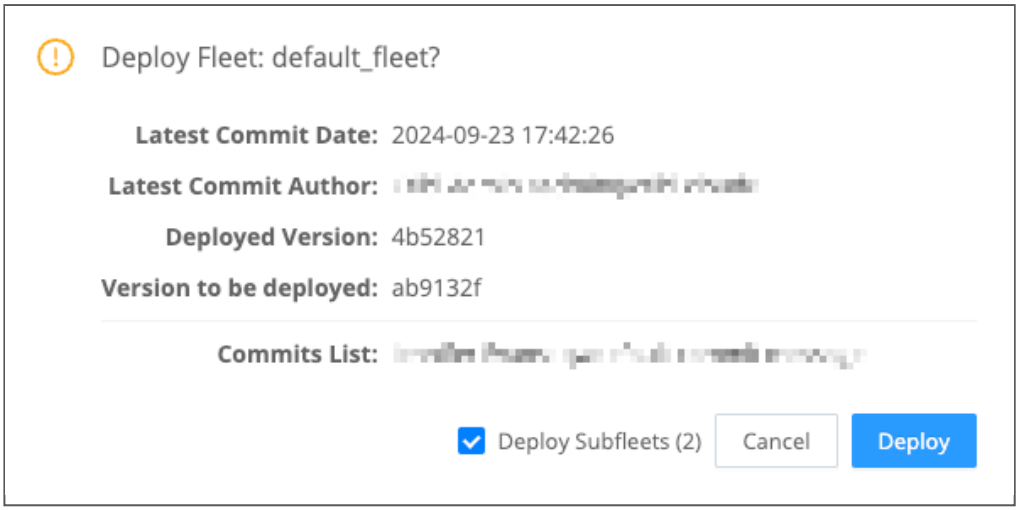
Once you save and commit your configuration changes, you can deploy the updated configs to the Edge Nodes. You deploy new configurations at the Fleet level.

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_MASTER_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_MASTER_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), `master` (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.

4.1.5. Recovering 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.

You can, however, still recover a deleted Fleet, using one of the following two methods:

- Create a new Fleet with the same name.
- Revert commit via command line.

Creating a New Fleet with the Same Name

When you create a new Fleet with the same name as a deleted one, it will restore that Fleet with all of its configuration.

1. Select **Fleets** from the sidebar, then **Add Fleet**.
2. Set **Fleet name** to be exactly the same as the deleted Fleet. Fill in the remaining information and **Save**.
3. If you previously committed the deletion of the original Fleet, open the **Version Control** menu.
4. Select a previous version of the deleted Fleet and confirm with **Done**.

Reverting Commit via Command Line

You can restore a deleted Fleet in version control 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.

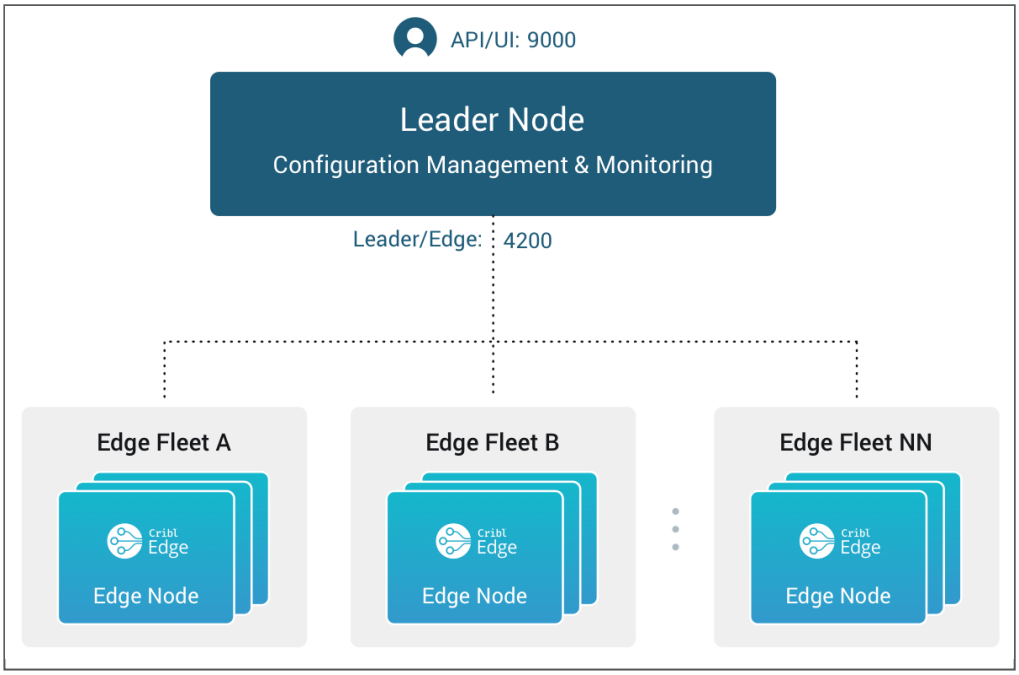
4.2. Edge Nodes

4.2.1. How Edge Nodes And Leader Work Together

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 (for example: live captures, teleporting, status updates, config bundle notifications, and so on).
- Edge Node to Leader Node: HTTPS 4200. Used for config bundle downloads.
- Edge Node bootstrapping/update: TCP 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, current software/configuration version, etc. – that the Leader tracks with the connection.

The failure of an Edge Node to successfully send two consecutive heartbeat messages to the Leader will cause the respective Edge Nodes to be removed from the Nodes page in the Leader's UI until the Leader receives a heartbeat message from the affected node.

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.

The Leader is unaware of Edge Nodes' platforms (i.e, Linux or Windows) within a Fleet. So the ConfigHelper omits platform-specific limitations. Therefore, when you manage Edge Nodes on heterogeneous platforms, create a Windows-specific Fleet and mapping. See Managing Edge Nodes on Multiple Platforms.

4.2.2. Manage Edge Nodes

How to manage Edge Nodes in a Fleet and perform tasks like teleporting and exporting a list of Nodes

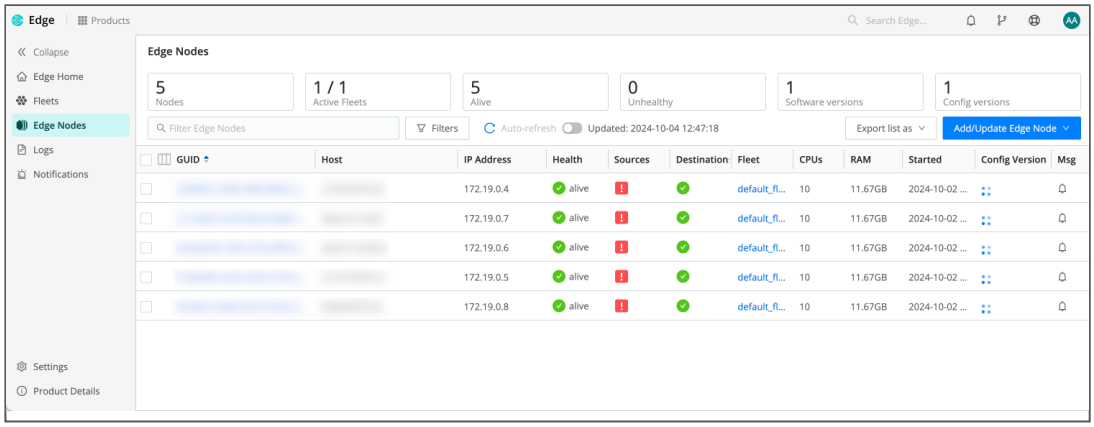
---

Accessing the **Edge Nodes** page is available on certain plan/license tiers. For details, see Pricing.

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 in the selected Fleet

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.

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.

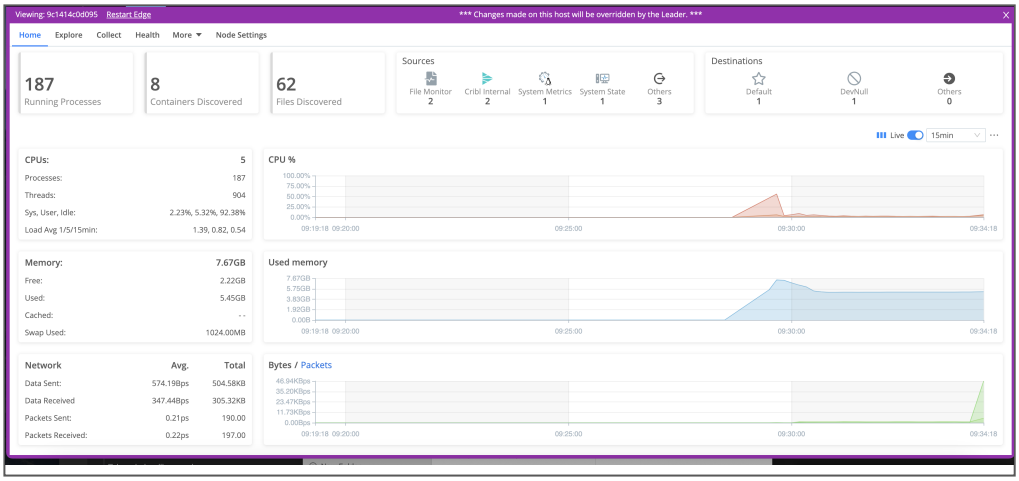
Teleport into an Edge Node

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.

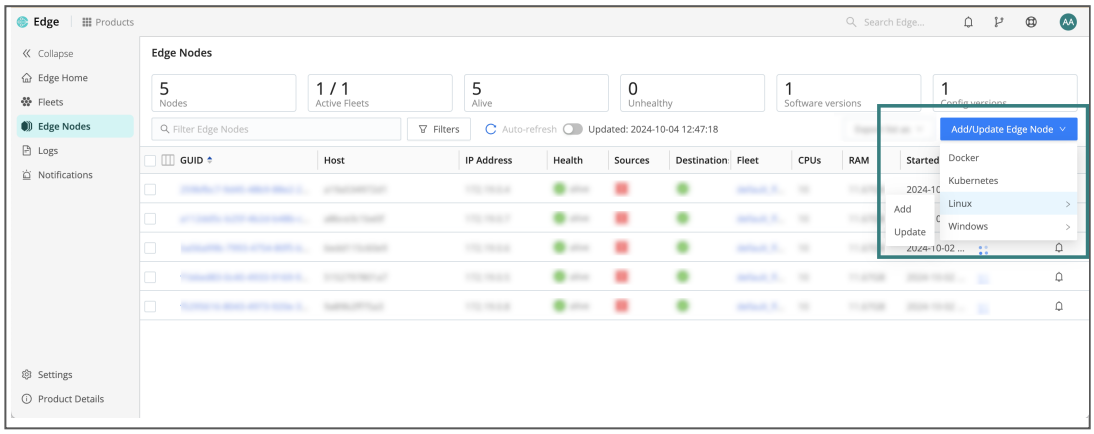
2. Navigate to **Node Settings** then **Distributed Settings**.
3. Enter tags for this Node and select **Save**.

In Cribl.Cloud, you must accept the confirmation modal. Changing the tags on an Edge Node can affect the Fleet that the Leader maps it into.

4.2.3. 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.

You can generate a bootstrap script for Edge Nodes deployed in the following environments:

- Docker: For details, see Running in a Docker Container.
- Kubernetes: For details, see Deploying via Kubernetes.
- Linux: For details, see Installing Cribl Edge on Linux.
- Windows: For details, see Installing Cribl Edge on Windows.

The **Update** option in the UI adjusts the Leader connection details for the currently installed software. To upgrade the Edge Node – that is, install a newer version of the software – see Upgrading.

Add a New Edge Node with a Bootstrap Script

Using the Cribl Edge UI, you can add Edge Nodes to your existing deployment. Simply 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.

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`.
- Port 443 to a Cribl.Cloud Leader.
- Port 9000 to an on-premises Leader.

How To Add an Edge Node

The details below differ slightly depending on which deployment option you select. This tutorial assumes a **Linux deployment**.

1. From the Cribl Edge sidebar, select **Edge Nodes**.
2. On the **Edge Nodes** page, select **Add/Update Edge Node**.
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 **Fleet**, as well as the **Leader hostname/IP** (URI).
    - As needed, correct the **User** and **User Group** to run Cribl as. Defaults to `cribl`.
    - As needed, correct the **Installation directory**. Defaults to `/opt/cribl`.
4. Optionally, add **Tags** that you can use for filtering and grouping in Cribl Edge. Use a tab or hard return between (arbitrary) tag names.
5. Copy the resulting script to your clipboard.
6. Click either **Done** or **X** to close the modal.
7. 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.

Select the Update Script Type for Linux Nodes

When you update a Linux 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 `systemctl edit cribl-edge` on the Edge Node.
    2. Add `ENVIRONMENT="CRIBL_DIST_MASTER_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_MASTER_URL=tcp://criblmaster@masterHostname:4200 ./cribl restart
```

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.

Concatenate the Bootstrap Script

Cribl Edge admins can use the UI to concatenate and copy/paste the bootstrap script, combining several of the steps above. Instead of manually editing the fields in the UI, you can change them in the script. You can use an adjacent option to grab a script that updates an Edge Node's Fleet assignment.

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, 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 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.

Troubleshooting the Display

If you see unexpected results in the **Edge Nodes** list, keep in mind that:

- Edge Nodes will be removed from the list when they miss five heartbeats or whose connections have been closed for more than 30 seconds.
- 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 has not yet 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.

4.2.4. 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.

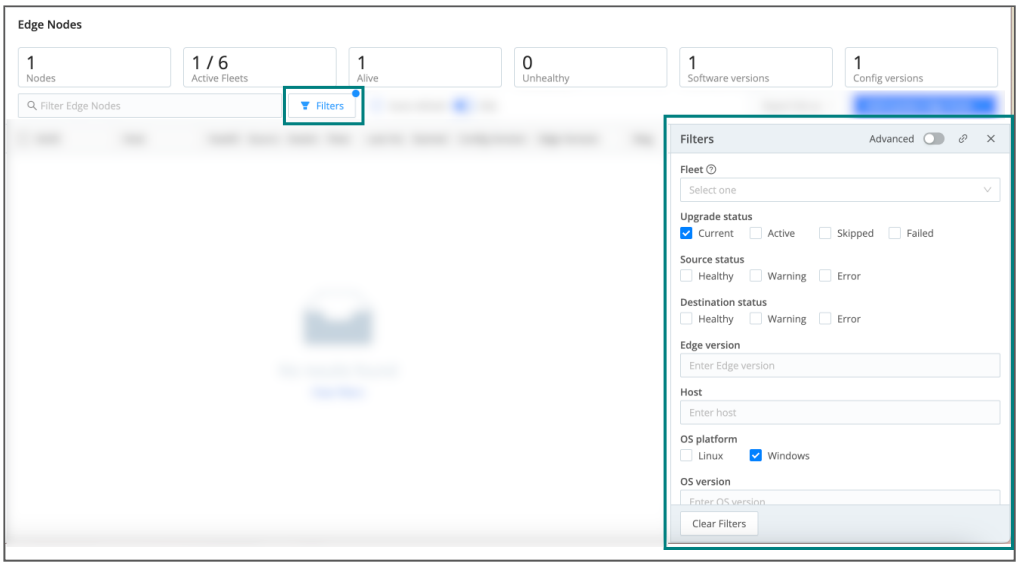
For example, you may want to view all of the Edge Nodes that are experiencing upgrade issues. Or, you may need to update the configuration of Nodes that are running a certain operating system version, but you don't know which Fleets they belong to.

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, in both List View and Map View. 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** 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 it.

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 on **Advanced** mode.

Field	Description	Javascript property
Fleet	Find Nodes based on their Fleet assignment. Visible when filtering from the **Edge Nodes** page.	`group=='<fleet>'`
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. • **Skipped**: Nodes with a skipped upgrade. • **Failed**: Nodes that failed to upgrade.	`nodeUpgradeStatus.state`
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>')`

Field	Description	Javascript property
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>')`
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 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 the following Node properties and types to create your Advanced filter in Javascript.

Node Property	Type	
id	string	
status	string (Healthy)	
group	string (Fleet name)	
info		
	hostname	string
	platform	string (OS Platform; *linux, win32 etc.*)

Node Property	Type	
	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 (epoch timestamp in msecs wh guid: string (unique instance ID) installType: string — Install Type. Can be CONTAINER If this is set, Leader upgrades are tags: string array (tags configured on the No version: string (Edge version)
	freeDiskSpace	number (Free disk space in bytes)
	totalDiskSpace	number (Total disk space in bytes)
	kube (this will only be populated for Nodes running in a Kubernetes Cluster)	source: string (always cluster) node: string (Node name where this Pod is ru pod: string (Pod Name) namespace: string (namespace, always cribl)

<table>
<tr><th>Node Property</th><th>Type</th></tr>
<tr><td></td><td>
<table>
<tr><td></td><td>
<table>
<tr><td>owner</td><td>
<table>
<tr><td>kind</td><td>string (DaemonSet)</td></tr>
<tr><td>name</td><td>string (cribl-edge)</td></tr>
</table>
</td></tr>
<tr><td>conn ip</td><td>string (this is the IP that Cribl Leader sees; different fro</td></tr>
<tr><td>localTime</td><td>number (local epoch time on the Node in msec)</td></tr>
</table>
</td></tr>
<tr><td>lastMsgTime</td><td>number (time when Node sent the last message in msec)</td></tr>
<tr><td>firstMsgTime</td><td>number (time when Node sent the first message in msec)</td></tr>
<tr><td>nodeUpgradeStatus</td><td>
<table>
<tr><td>timestamp</td><td>number (time when Node entered the state below in ms</td></tr>
<tr><td>state</td><td>number
0 for Current (Node is upgraded to version setting in the
1 for Active (Node is being upgraded to version setting i
3 for Failed (upgrade failed when tried to upgrade to ver</td></tr>
<tr><td>skipped</td><td>number (if set, this Node is skipped by thr Leader's auto</td></tr>
</table>
</td></tr>
<tr><td>lastMetrics</td><td>
<table>
<tr><td>edge.dropped events</td><td>number (events dropped by Node)</td></tr>
<tr><td>health.inputs</td><td>number (Source health status)
0 - All sources in the Node are healthy
1 - One or more sources in the Node have warnings
2 - One or more sources in the Node have errors.</td></tr>
<tr><td>health.outputs</td><td>number (Destination health status)
0 - All sources in the Node are healthy
1 - One or more sources in the Node have warnings
2 - One or more sources in the Node have errors</td></tr>
<tr><td>total.in_events</td><td>number (total events ingested by Node)</td></tr>
<tr><td>total.out_events</td><td>number (total events sent out by Node)</td></tr>
</table>
</td></tr>
</table>

Node Property	Type	
	total.dropped_events	number (total events dropped by Node)
	total.in_bytes	number (total ingested bytes)
	total.out_bytes	number (total bytes egressed)

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
```

Nodes running the System State Source

```
lastMetrics.hasOwnProperty('health.inputs:input:system_state:in_system_state')
```

Nodes Running the System State Source and experiencing issues with Destinations

```
lastMetrics.hasOwnProperty('health.inputs:input:system_state:in_system_state') && las
```

Nodes with errors in either Sources or Destinations

```
lastMetrics['health.inputs'] === 2 || lastMetrics['health.outputs'] === 2
```

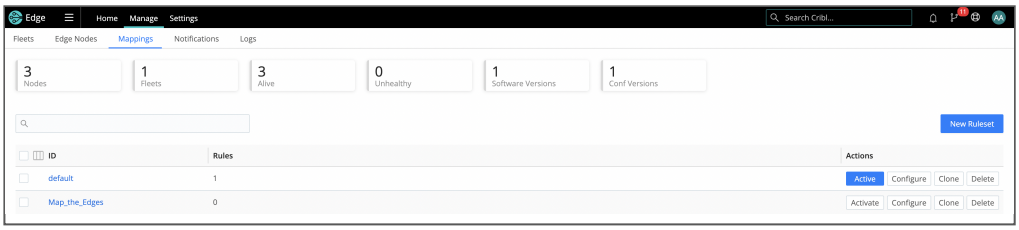
4.2.5. Mapping Edge Nodes To Fleets

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

Click the **Mappings** tab (**Manage** > **Mappings**) to display status and controls for the active Mapping Ruleset.



Mappings status/controls

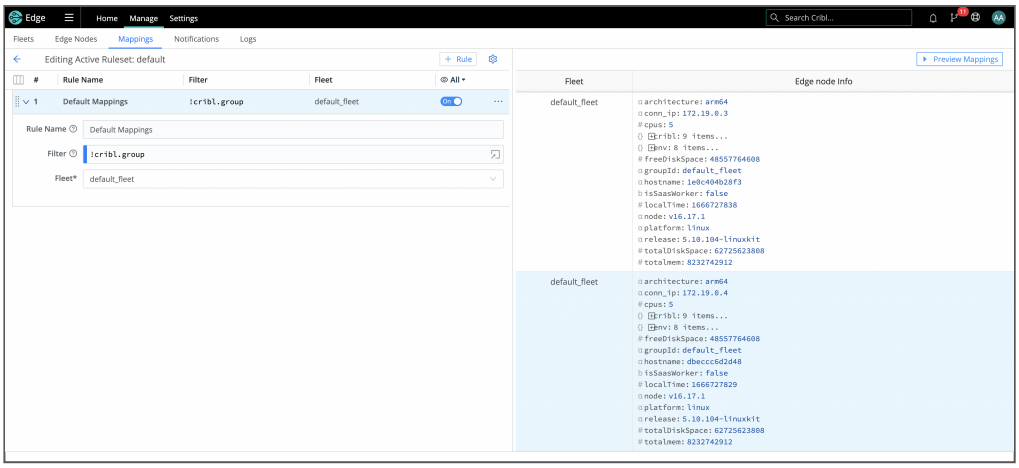
This page displays a maximum of 10,000 results.

Mapping Rulesets

Within a Ruleset, a list of rules evaluate Filter expressions on the information that Edge Nodes send to the Leader.

To manage and preview the Rules in a Ruleset, click into it.

In a Ruleset, the order of Rules 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. system.

Creating a Mapping Ruleset

To create a Mapping Ruleset:

1. Select **Manage** on the top nav > **Mappings**.
2. Select **Add Ruleset**.
3. Give the **New Ruleset** a unique **ID** and click **Save**.
4. Select **Configure** and start adding rules with **Add Rule**.

As you build and define rules, the Preview pane will display all the Edge Nodes that will map to Fleets matching your defined rules.

You must activate a ruleset before the Leader can use it. To activate it, go to **Mappings** and click **Activate** on the required ruleset. The **Activate** button will then change to an **Active** toggle. Using the adjacent buttons, you can also **Configure** or **Delete** a ruleset, or **Clone** a ruleset if you'd like to work on it offline, test different filters, etc.

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.

For platform-specific mapping, set the `platform` property to `win32` for Windows, or to `linux` for Linux.

Example Mapping Rule

Within a Mapping Ruleset, click **Add Rule** to define a new rule. 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:
- More than 6 CPUs OR `CRIBL_HOME` environment variable contains `DMZ`, AND:
- Belongs to `Fleet420`.

Rule Configuration

- **Rule Name**: `myFirstRule`
- **Filter**: `(conn_ip.startsWith('10.10.42.') && cpus > 6) || env.CRIBL_HOME.match('DMZ')`
- **Fleet**: `Fleet420`

Default Fleet and Mapping

A Cribl Edge instance will create the following automatically when it runs as Leader:

- A `default_fleet` Fleet.
- A `default` Mapping Ruleset,

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.
- Else, map to the `default_fleet` Fleet.

Managing Edge Nodes on Multiple Platforms

The Leader is unaware of Edge Nodes' platforms, such as Linux or Windows, within a Fleet, which means the ConfigHelper omits platform-specific limitations. Therefore, when you manage Edge Nodes on heterogeneous platforms, create a Windows-specific Fleet and mapping.

For what's supported on Windows, see Cribl Edge on Windows.

Collecting AWS EC2 Tag Metadata

If Edge is connected to an AWS EC2 instance, the **Mappings** tab can gather metadata for AWS instance tags. You can use tags to map EC2 nodes to the correct Fleet.

You must allow access to tags in the EC2 instance metadata for Cribl Edge to obtain them.

4.2.6. Managing Edge Node Config Bundles

Learn about Edge Node config bundles and how to manage them

Config bundles are compressed archives of all config files and associated data that an Edge Node needs to operate. 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.

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.

Bundle in S3

You can store bundles in an AWS S3 bucket to reduce the Leader load. 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**: When configuring **Leader Settings** on a Leader Node, specify an S3 bucket (format: `s3://${bucket}`) for remote bundle storage in the **S3 Bundle Bucket URL** field.
- **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 adding the S3 access (`*.s3.amazonaws.com`) to the allowlist in your firewalls or proxies (due to non-static S3 addresses). This proactive step helps avoid disruptions.

This multi-step approach ensures efficient and reliable Edge Node updates in various network configurations.

4.2.7. Set Up Leader And Edge Nodes

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

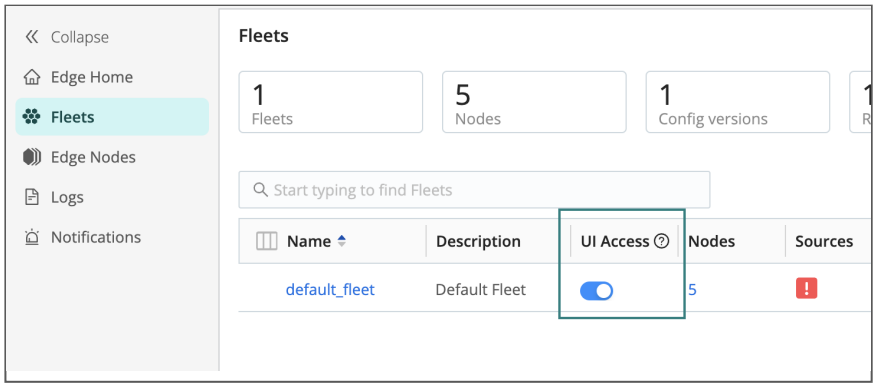
1. Navigate to **Settings** > **Global** > **Distributed Settings** > **General Settings**.
2. Select **Mode**: **Leader**.
3. Select the **Leader Settings** menu.
4. Confirm or enter the required Leader settings (**Address** and **Port**). Customize the optional settings if desired.
5. Select **Save** to restart the Cribl Edge server.

Edge Node UI Access

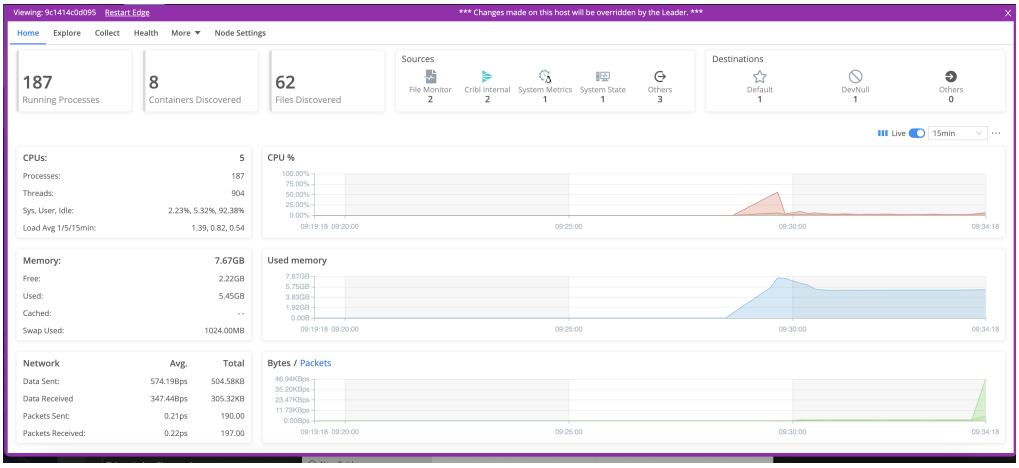
This useful option enables you to click through from the Leader's **Edge Nodes** page to an authenticated view of each Node's UI. We call this Teleporting into an Edge Node. The instructions below correspond to enabling the `groups.yml` file's `workerRemoteAccess` configuration key.

To enable Node UI access from the Leader's UI:

1. Select **Fleets** from the sidebar.
2. For each desired Fleet, toggle on **UI Access**.



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.yml` on Windows), under the `distributed` section, set `mode` to `master`:

$CRIBL_HOME/local/_system/instance.yml or C:\ProgramData\Cribl\local_system.yml

```
distributed:
  mode: master
  master:
    host: <IP or 0.0.0.0>
    port: 4200
    tls:
      disabled: true
    ipWhitelistRegex: /.&ast;/
    authToken: <auth token>
    enabledWorkerRemoteAccess: false
    compression: none
    connectionTimeout: 5000
    writeTimeout: 10000
```

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:
`systemctl restart systemd-tmpfiles-clean.service`

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 relatively close to the root; and the Cribl admin must have user permissions to write to it. As one example of a protected directory, you could specify: `/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.yml` on Windows), under the `distributed` section, set `mode` to `managed-edge`:

```
$CRIBL_HOME/local/_system/instance.yml or C:\ProgramData\Cribl\local\_system.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
```

Use Environment Variables

You can configure Edge Nodes via environment variables, as in this example:

```
CRIBL_DIST_MASTER_URL=tcp://${CRIBL_DIST_TOKEN:-criblmaster}@masterHostname:4203
./cribl start
```

For additional details, see 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]
```

The command requires the `-H` and `-p` parameters. For other options, see the CLI Reference. Here is an example command:

```
./cribl mode-managed-edge -H 192.0.2.1 -p 4200 -u myAuthToken
```

Edge will need to restart after you issue this command.

5. Cribl Edge Better Practices

5.1. Configuring Cribl Edge

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 Fleets Hierarchy and Design.

Configuring multiple Fleets is available on certain plan/license tiers. For details, see Pricing.

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 **Age duration limit** 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 hosts. While the option will still appear in the UI for Edge Fleets that include Windows 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 **Age duration limit** 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 **Age duration limit** 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 **Age duration limit** 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.

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 license 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 impact on Edge Node performance. Some Packs might generate significant CPU load, especially those with large lookup tables or complex processing Pipelines. So it's generally recommended to keep Edge agents as

streamlined as possible. By carefully evaluating the benefits and trade-offs of each Pack, you can optimize your Cribl Edge deployment for both functionality and efficiency.

5.2. Fleets Hierarchy And Design

Cribl Edge enables you to flexibly group Edge Nodes into logical Fleets and Subfleets, organized as you like. However, Fleet design is extremely important for the health of the Leader. In this guide, we address "better practices" for designing your Fleets and Subfleets.

Leader Health

Below are some guidelines for designing your Fleets to optimize your Leader's health:

- 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 a larger number of Edge nodes, change the default metrics controls to `minimal`.

Fleet Design Better Practices for Enterprises

Consider the following "better practices" for designing Fleets for enterprises:

- Do not use `default_fleet`. The `default_fleet` is the fallback for mapping errors. If you purchased an Enterprise License, disable all the Sources in the `default_fleet` to prevent accidental collections. This guideline doesn't apply to the free license, where the only option is `default_fleet`.
- Design your Fleet hierarchy before deploying your Edge Nodes. For details, see Designing a Fleet Hierarchy below.
- Use the bootstrap script in the Add/Update Edge Node modal to deploy Edge Nodes to a specific Fleet. Depending on your Fleet design, we recommend adding tags to the bootstrap script to uniquely identify the Fleet. You can use these tags for automatic mappings.
- 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`. If you disable all the Sources in the `default_fleet`, there will be no impact on data consumption.
- Monitor your `default_fleet` for any unintentional/rogue Edge Nodes.

Designing Your Fleet Hierarchy

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` > `Application`. In this case, the Edge Nodes at the `Application` level can inherit and share configurations from the `Org` 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.

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, etc.): Another good way to organize your Fleets can be based on server functions.
- OS-based: If you are running an environment with both Linux and Windows machines, consider separating them into different Fleets per OS.

The recommendations here:

- To avoid confusion, try to keep Fleets to two to three levels. We'll discuss it more under Inheritance Considerations.

To learn more about Fleets, see Fleets and check out Configuring a Fleet for configuration details.

Naming your Fleets

What's in a name, you ask? Well, a whole lot when it comes to Fleets. 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 four-level Fleet in an intuitive way:

- BZT
    - BZT-NewYork
        - BZT-NY-Linux
            - BZT-NY-Linux-Apache
            - BZT-NY-Linux-Gizmotrax

        - BZT-NY-Windows
            - BZT-NY-Win-IIS
    - BZT-LA
        - BZT-LA-K8s
        - etc.

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.

Inheritance Considerations

As suggested above in Designing Your Fleet Hierarchy, plan and design the Fleets' inheritance in advance. 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. Consider:

- Applying certificates and other company-wide settings at this level.
- Avoid OS-specific or site-specific configurations at this level.

Subfleet 1 (Parents)

Per-site configurations. Consider:

- Configure site-specific Sources/Destinations at this level.
- Specify any site-specific configurations, like common Destinations.

Subfleet 2 (Children)

Configurations used by groups of similar systems. Consider:

- 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.

Subfleet 3 (Grandchildren)

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

You can add up to four levels of Subfleets underneath the parent Fleet, and each Subfleet will inherit the configurations of every parent Subfleet before it.

If you make a change to an inherited config (such as Sources/Destinations) in a Subfleet, the Subfleet will no longer get updates made to the parent Fleet for that inherited config. Other configs are still inherited from that parent.

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.

Finally, note that Subfleets don't inherit Packs and Pipelines from a parent Fleet at this time. You must recreate or import them into child Fleets.

5.3. Monitoring Your Infrastructure With Cribl Edge

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

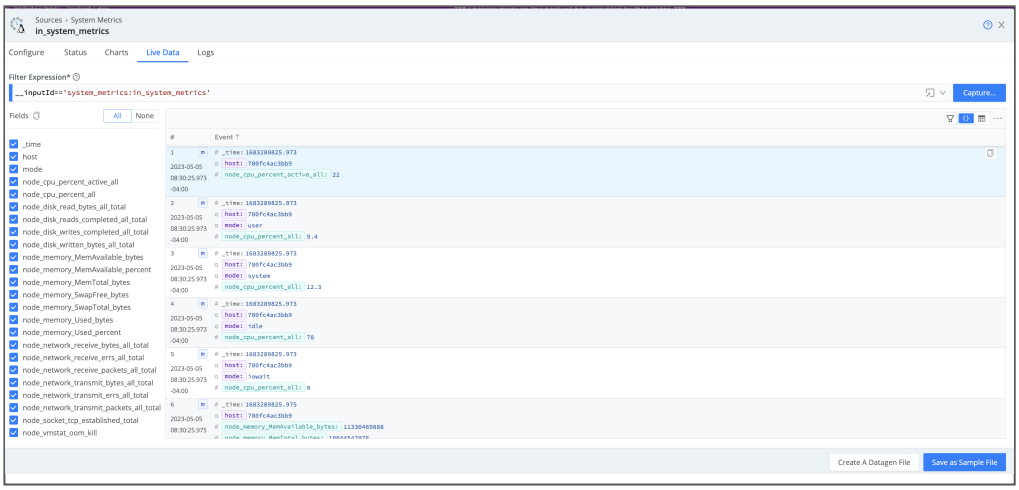
2. Decide what level of metrics detail you want to collect and display.

In the System Metrics Source and Windows Metrics Sources, go to the **Host Metrics** and **Container Metrics** tabs:

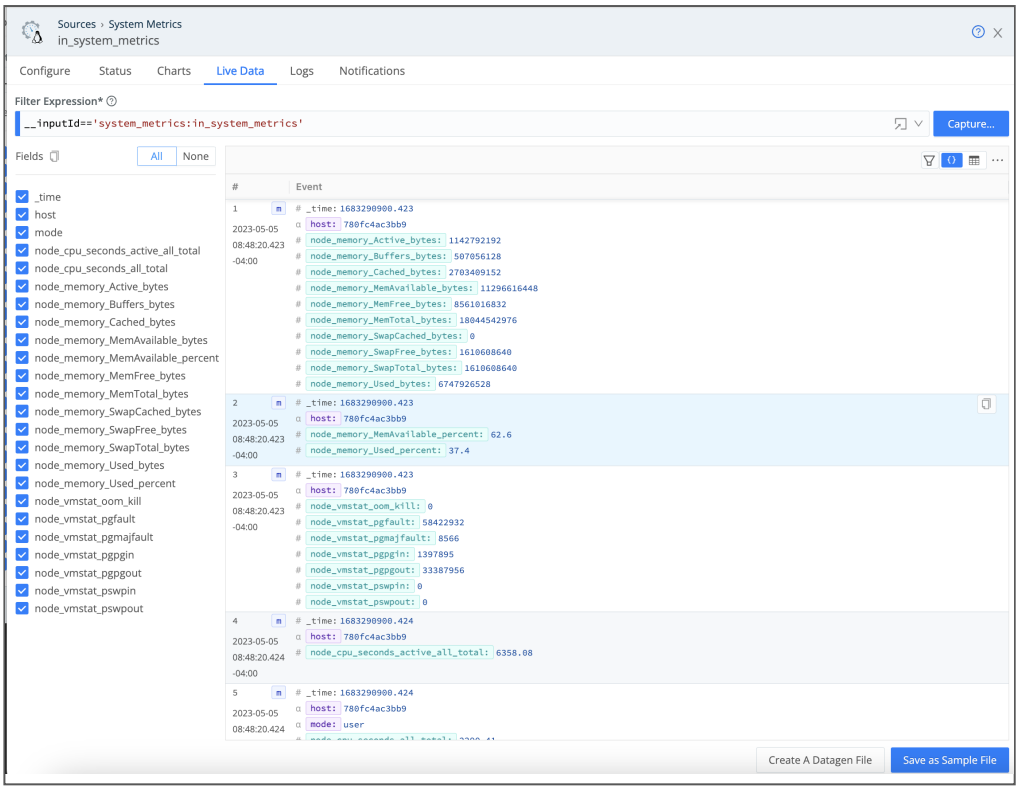
- For the relatively simple dashboard showing aggregate metrics, choose the **Basic** mode for most of your metrics.
- For the highly detailed metrics visualizations dashboard, choose the **All** mode for all of your metrics.

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

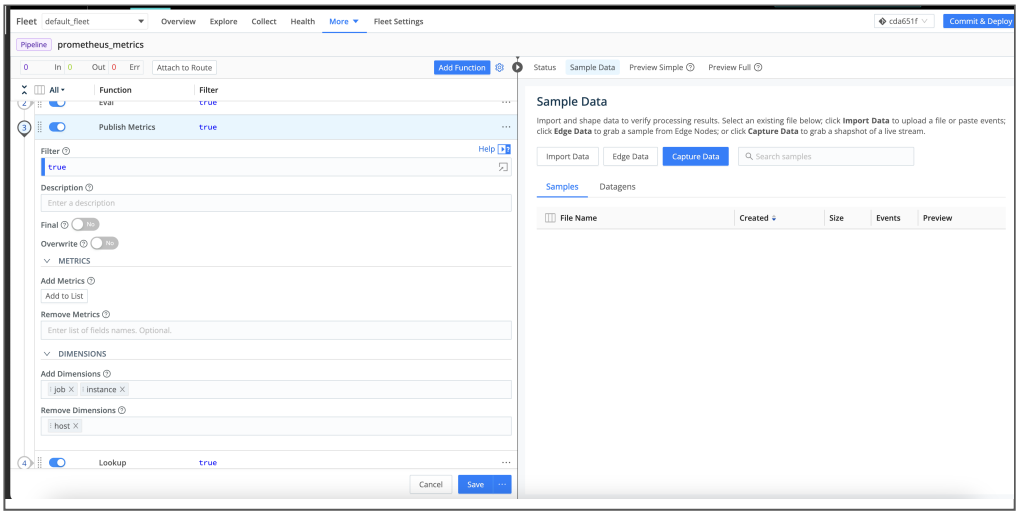


Detailed System Metrics

(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.

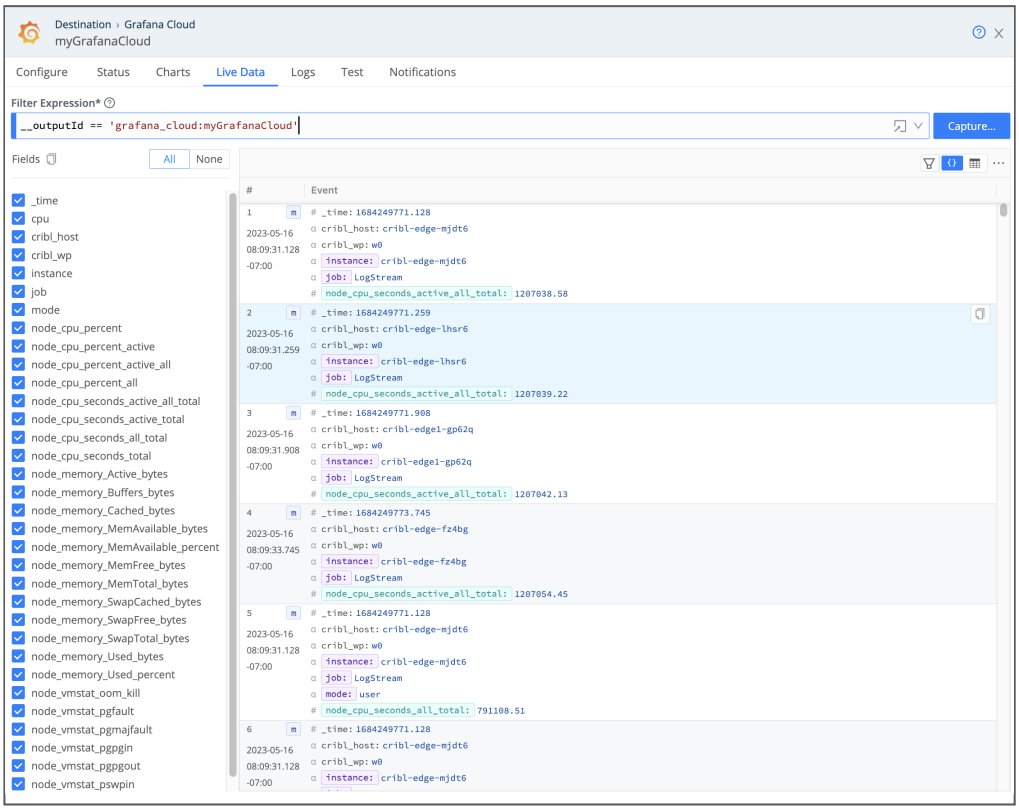


Customizing the Pipeline's dimensions

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.

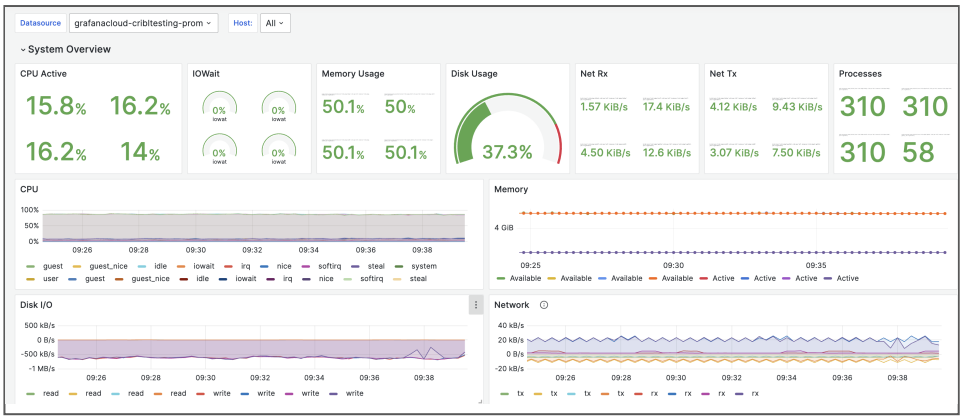


Preview Live Data

Visualize the Data in Grafana

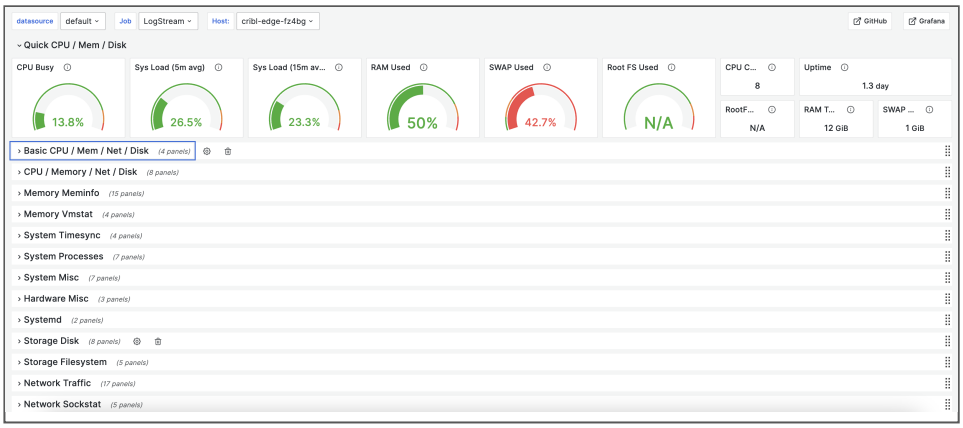
You can now visualize metrics in Grafana Cloud. To get started:

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 recommend configuring your System Metrics Source (as discussed above) to receive **All** metrics, and then using the Node Exporter Full dashboard below for visualization.



Node Exporter Full

5.4. Kubernetes Observability Using Cribl Edge

Capturing observability data is difficult in Kubernetes environments. Kubernetes deployments can scale up to thousands of nodes, making monitoring challenging. In addition, tracking and troubleshooting infrastructure performance is challenging due to its dynamic nature.

In this guide, we will show you how to address these problems by:

- Deploying Cribl Edge via Kubernetes using the Cribl Helm Chart, including creating Kubernetes-specific Fleets and > Fleet settings.
- Setting environment variables for your Kubernetes environment to work.
- Configuring the Kubernetes Sources, Destinations, and Pipelines to connect them.
- Visualizing the data in Grafana using some pre-built dashboards.
- Learning about “better practices” and recommendations for deploying Cribl Edge on your Kubernetes clusters.

While some open-source tools can address these needs, they may be cumbersome to deploy and manage for sprawling microservices environments. Getting Kubernetes data into analytics platforms is challenging due to its volume, lack of standard formatting, and ephemeral nature.

In addition to addressing these issues, Cribl Edge brings Cribl Stream’s processing power for deployment on servers. With the ability to collect, process, and forward data with low resource overhead, Cribl Edge is designed for sprawling environments such as Kubernetes. With Cribl Edge, you can capture Pod logs and Control Plane events, generate Kube State Metrics (KSM), scrape data from Prometheus endpoints, and send the data to your downstream systems.

Deploy Cribl Edge via Kubernetes

In this section we’ll cover:

- Configuring Fleets and Fleet Settings
- Installing Using Helm
- Setting Environment Variables for your Kubernetes Deployment

Configure Fleets & Fleet Settings on Cribl Edge

Before we deploy the Edge Node, let’s configure a Fleet. Fleets and their corresponding Subfleets allow you to author and manage configuration settings for a particular set of Edge Nodes.

We recommend configuring a separate Fleet to manage your Kubernetes environment. It is easier to deploy or upgrade Kubernetes Edge Nodes using our Helm Charts when they are all in one Fleet. For best practices on configuring Fleets, see Fleet Design and Hierarchy.

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.

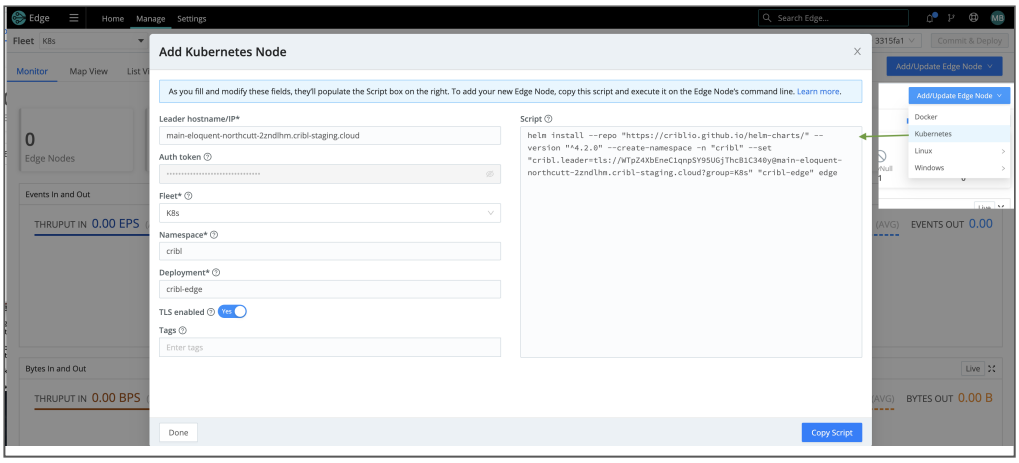
Use the Bootstrap Script to install the Helm Chart

We recommend deploying the Cribl Edge agent in your Kubernetes cluster using 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.

Next, bootstrap a new Edge Node on your Kubernetes Fleet using the Helm Script found in the UI. Copy/paste the bootstrap script for adding a Kubernetes node.

Below is a composite screenshot of adding an Edge Node via Kubernetes.



Generating a script to bootstrap an Edge Node on Windows

See Deploying via Kubernetes for details on installing Cribl Edge.

Environment Variables and Metadata

You can customize environment variables and metadata to modify (respectively) your Edge Nodes' deployment and reporting.

Environment Variables

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

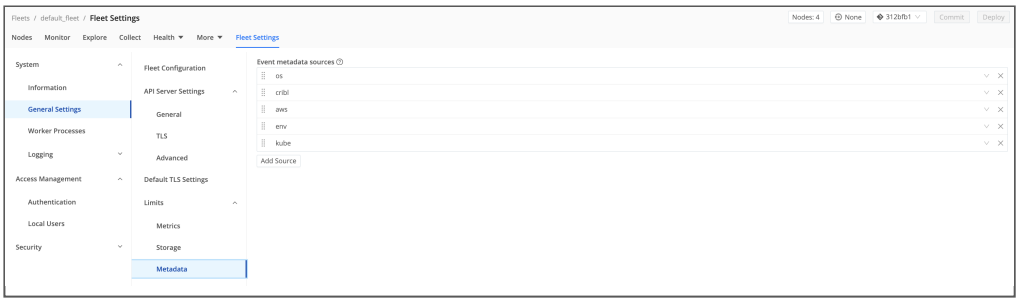
- `CRIBL_SERVICEACCOUNT_PATH` – by default, set to: `/var/run/secrets/kubernetes.io/serviceaccount`.
- `CRIBL_K8S_POD` – the name of the Kubernetes Pod in which Cribl Edge is deployed.
- `KUBECONFIG` – set this for local testing.
- `CRIBL_K8S_TLS_REJECT_UNAUTHORIZED` – set to `0` to disable certification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

For details on other environment variables you can set for your Kubernetes deployment, see Environment Variables.

Metadata

You can customize the type of metadata collected at **Fleet Settings** > **Limits** > **Metadata**.

In Edge mode, all the Event metadata Sources are enabled by default. Leave the `kube` metadata set to allowed (by default).



Set limits

The `kube` metadata (`__metadata.kube property`) reports details on a Kubernetes environment, including the node, Pod, and container.

For the `__metadata.kube property` to report details on a Kubernetes environment, some level of Kube API access is required if you want to use this metadata property.

`KUBE_K8S_POD`, an environment variable in the Cribl Helm Chart, is set by default to the name of the Pod where Cribl Edge is running by default. This adds the `__metadata.kube` property with information to report on the Nodes and Pods.

Configure the Kubernetes Sources

For the Kubernetes Sources to work as designed, Cribl recommends deploying Cribl Edge as a Daemonset.

Kubernetes Logs Source

The Kubernetes Logs Source connects to the Kubernetes API and loads the lists of Pods on the node, on a configurable 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 we can collect them.

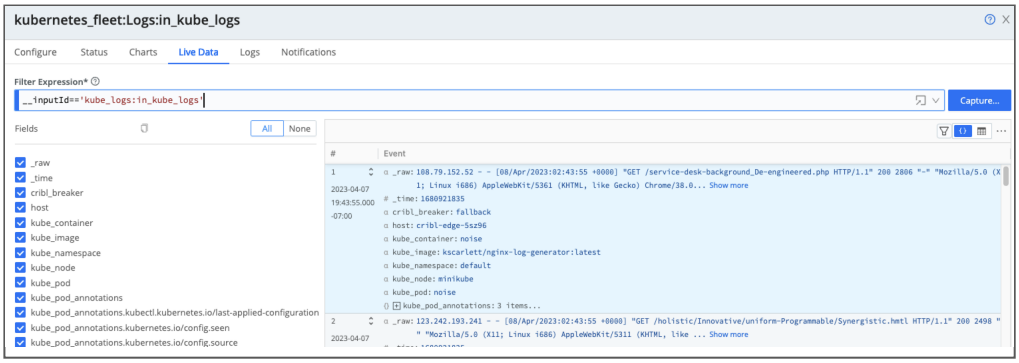
For example, if the container emits `1 MBps` and rotates logs at `10 MB`, the polling interval must be less than `10s`, to prevent hitting the `10 MB` rotation threshold twice in 10 seconds.

The Source then runs the Pods through the Filter Rules to determine which ones to report on. For example:

- Ignores Pods in the `kube-*` namespace - !metadata.namespace.startsWith('kube-')
- Ignore all DaemonSets – metadata.ownerReferences[0].kind != 'DaemonSet'
- Ignore Pods with specific Container names – spec.containers[0].name != 'edge'

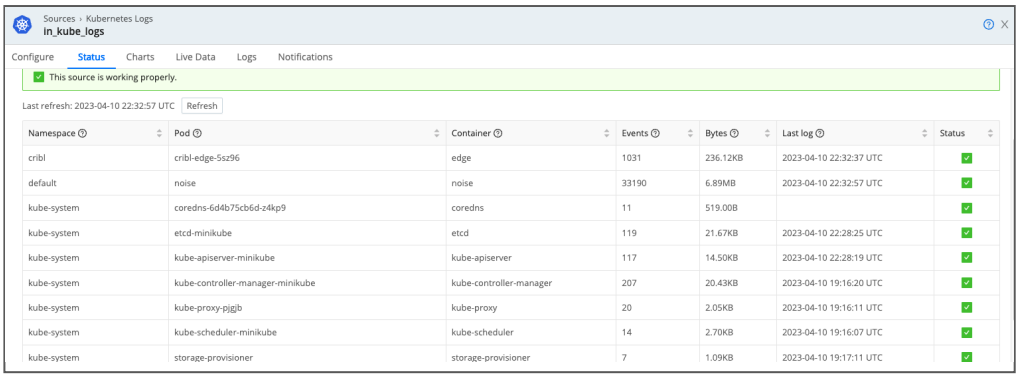
Configure the Kubernetes Logs Source to collect the Standard output (`stdout`) and Standard error (`stderr`) log streams from each Pod deployed inside the cluster.

Optionally, you can filter and enrich incoming logs.



Live Data: Kubernetes Logs Source

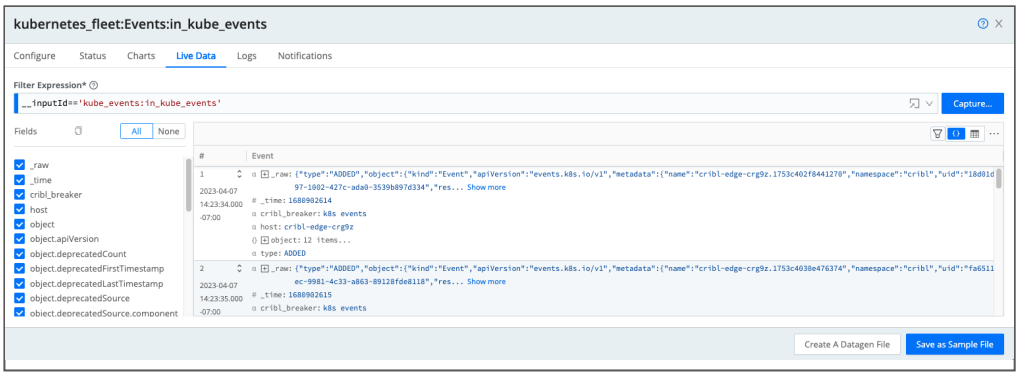
Below the **Capture** button, click the ellipsis (three dots or **...**) and then enable the **Show Internal Fields** toggle to explore the `__metadata` fields forwarded from the Cribl Edge Nodes.These metadata fields show information about the Pod, Node, and Cribl environment where they were collected.



Status: Kubernetes Logs Source

Kubernetes Events Source

Configure the Kubernetes Events Source to monitor state changes (such as Pod starts and jobs that fail to start) occurring on the Kubernetes cluster regarding workloads.



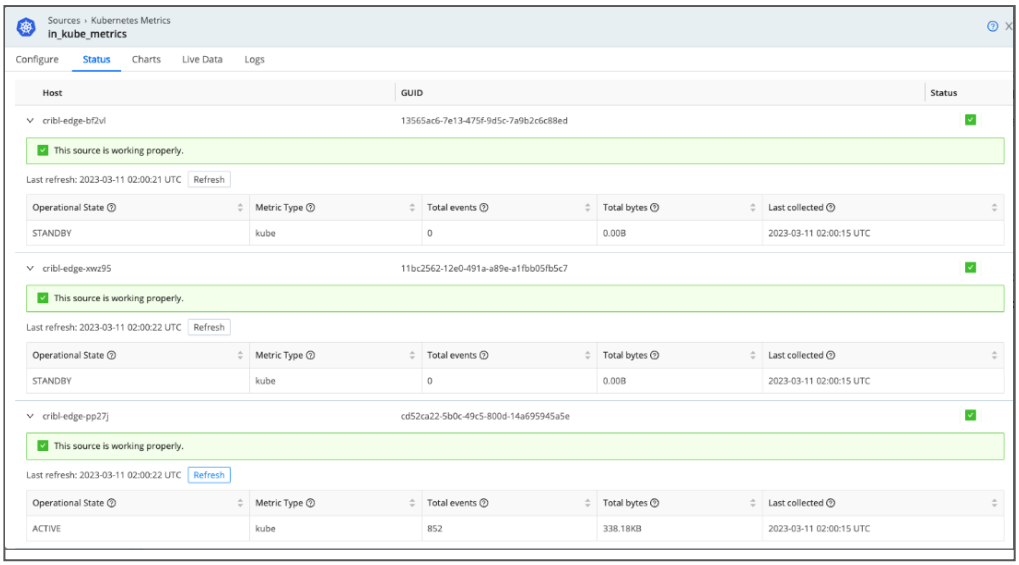
Live Data: Kubernetes Events Source

Kubernetes Metrics Source

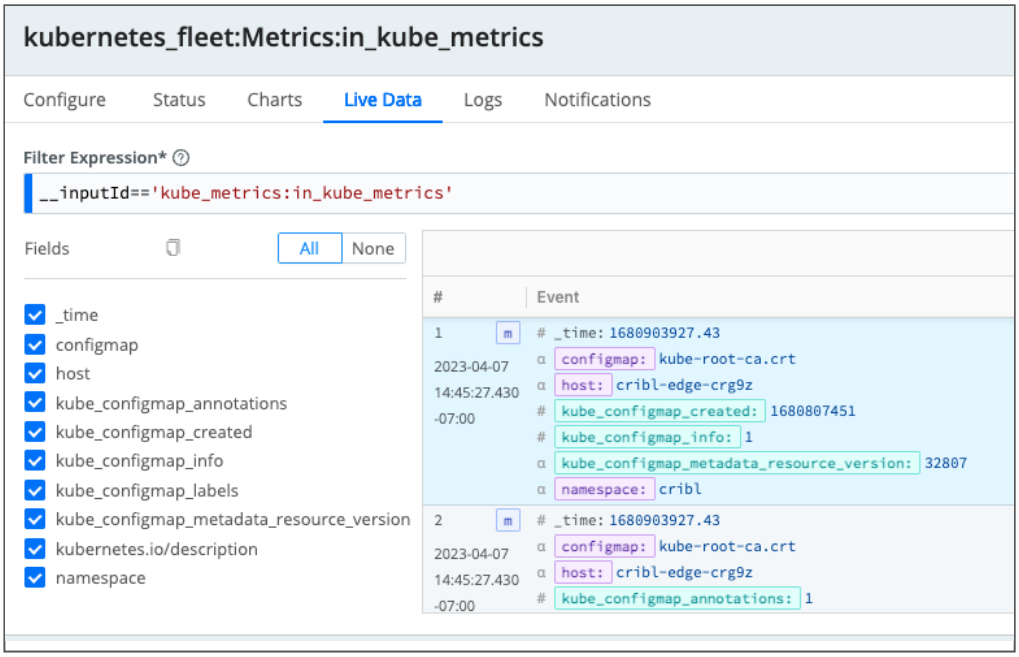
Configure the Kubernetes Metrics Source to generate metrics periodically based on the status and configuration of a Kubernetes cluster and its Nodes, Pods, and containers.

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 the Source's operational state, go to **Status** and expand the host details. The **Operational State** column shows either an **Active** or **Standby** state. **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



Live Data: Kubernetes Metrics Source

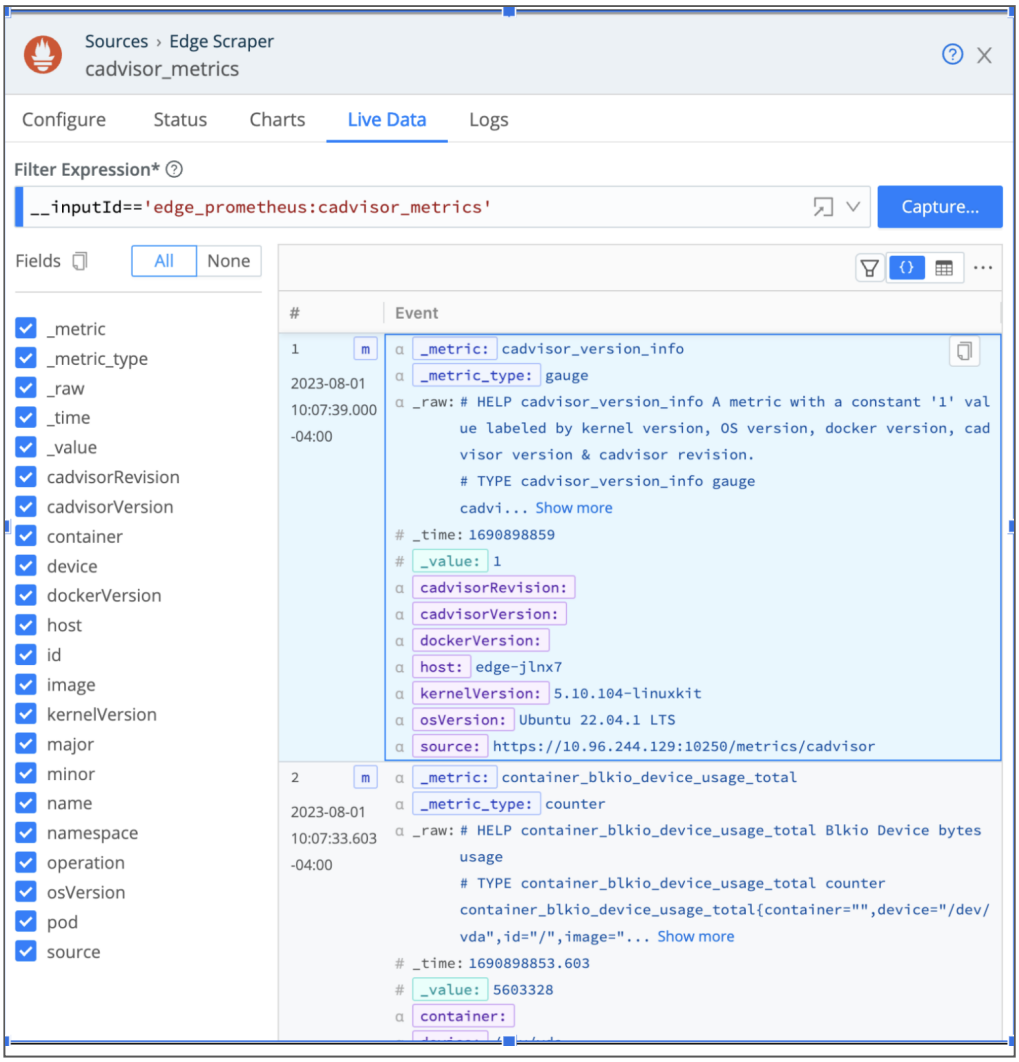
Prometheus Edge Scraper

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 us a really good idea of what's going on on each Node and Pod. In addition, this Source supports Disk Spooling and allows Cribl Search to query the metrics.

Using the Prometheus Edge Scraper, you can discover and collect metrics in Kubernetes Clusters dynamically. There are two Kubernetes-specific modes of operation:

- **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.
- **Pods**: When configured for Pod scraping, a single Edge Node will be elected to scrape the metrics from the discovered endpoints.



Live Data: Prometheus Edge Scraper

Configuring Your Destinations

For this example, we're going to use Prometheus and Loki for receiving metrics, logs, and events 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.

Configure the Loki Destination to send log events to Grafana's Loki log aggregation system.

Similarly, Prometheus uses names, labels (dimensions), and numerical values. You can then create dashboards to visualize the change in data points over time using these time series.

Configure the Prometheus Destination to forward the Kubernetes Metrics to a Grafana dashboard (for our purposes).

Configuring the Pipelines

We will create two Pipelines to process the data as it routes to the configured Destinations.

Connecting Logs and Events

As previously mentioned, 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. Configure a Pipeline with an Eval Function to add the labels to the `__labels` field.

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}`.

Once you create a Pipeline, add an Eval Function with the following config in the **Evaluate Fields** section:

Name	Value Expression
`__labels`	`{}`
`__labels.namespace`	`kube_namespace`
`__labels.pod`	`kube_pod`
`__labels.container`	`kube_container`

Next, in the Fleet's submenu, go to **Collect** to attach Kubernetes Logs to Loki using the configured Pipeline

You can connect the Kubernetes Events Source directly to your configured Loki Destination with the default `Passthru` Pipeline connection.

Connecting Metrics

You can modify the default `prometheus_metrics1` Pipeline to change a few metadata fields and use it to connect to the Prometheus Destination.

In the `prometheus_metrics1` Pipeline, expand the Eval Function with the following config in the **Evaluate Fields** section:

Name	Value Expression
`job`	job || 'cribl'
`instance`	__metadata.kube.node.metadata.name || host

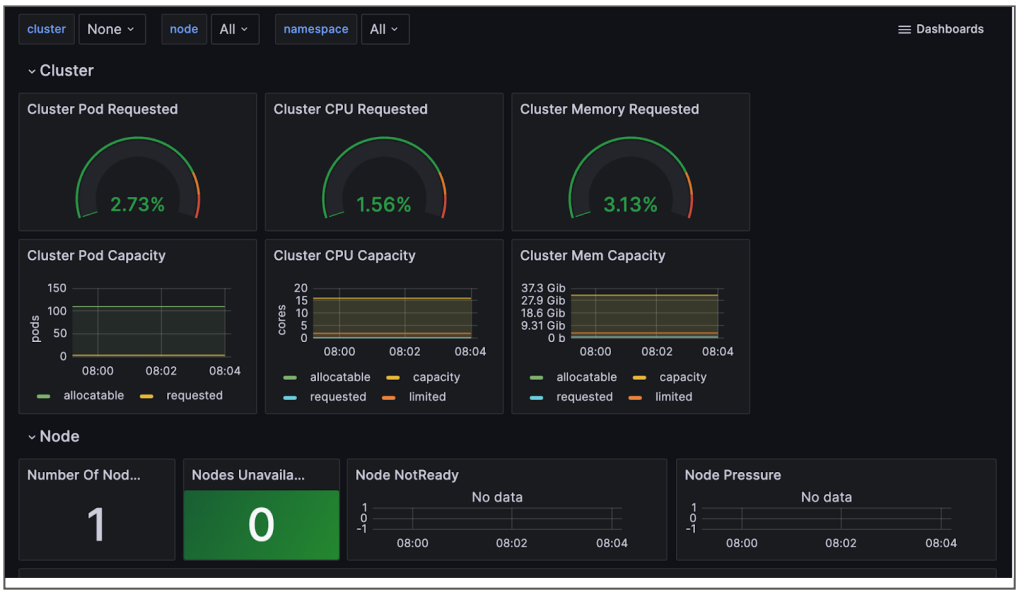
Next, in the Fleet's submenu, go to **Collect** to attach the Kubernetes Metrics Source (or Prometheus Edge Scraper) to the Prometheus Destination, using the `prometheus_metrics1` Pipeline as the connection.

Visualize the Data in Grafana

Navigate to your Grafana Cloud instance, and select Explore on the left panel to review the Node metrics. Selecting metrics and running the query will produce a chart and table of metric events that can be added to new or existing dashboards for analysis.

If you are missing some expected metrics, you might need to adjust the variables in your Grafana Cloud instance. For details, see Grafana's Variables topic.

Use Grafana's Filter option to browse the dashboards for your Prometheus data and Loki Logs. Below is a sample dashboard.



Grafana Dashboard

5.5. Windows Observability Using Cribl Edge

Cribl Edge offers holistic support for Windows observability, allowing comprehensive data collection across your Windows servers to provide reliable monitoring and analysis. You can run Cribl Edge on any supported Windows version to collect observability data (both metrics and logs) and send them to your desired destinations. By combining Cribl Edge with your favorite system of analysis, or data visualization tool, you can identify and investigate issues faster and monitor applications more effectively.

In this guide, we will walk you through the following:

- Deploy Cribl Edge on Windows by using the bootstrap script to add an Edge Node.
- Create a Windows-specific Fleet, and modify Mapping Rulesets to assign Nodes to Fleets.
- Explore the Windows Node and get a snapshot of the host system's current state.
- Locate and collect custom Windows event logs and send them to your desired destination.
- Collect Windows metrics and send them to your visualization tool of choice (or any supported destination).
- Collect logs using the File Monitor Source.

As a bonus, we'll walk you through how to use Cribl's Exec Source to execute Powershell commands and process them in Cribl.

Supported Windows Versions

Cribl Edge supports these versions of Windows:

Version	Workstation Type	Notes
Windows 10 and 11	Always-on network connection to the Leader.	Support for Windows workstations that do not have power management features enabled (sleep or hibernate).
Windows 10 and 11	**(Preview Feature)** Laptop or desktop with power management enabled.	Keep these things in mind when deploying Cribl Edge on Windows 10 and 11 workstations that are **not** always on: • 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.

Version	Workstation Type	Notes
		• HTTP Destinations might contain duplicate events. • Excessive logging has been observed when a Leader is unavailable to receive heartbeat messages from an Edge Node.
Windows Server 2016	Any	N/A
Windows Server 2019	Any	N/A
Windows Server 2022	Any	N/A

Deploy Cribl Edge on Windows

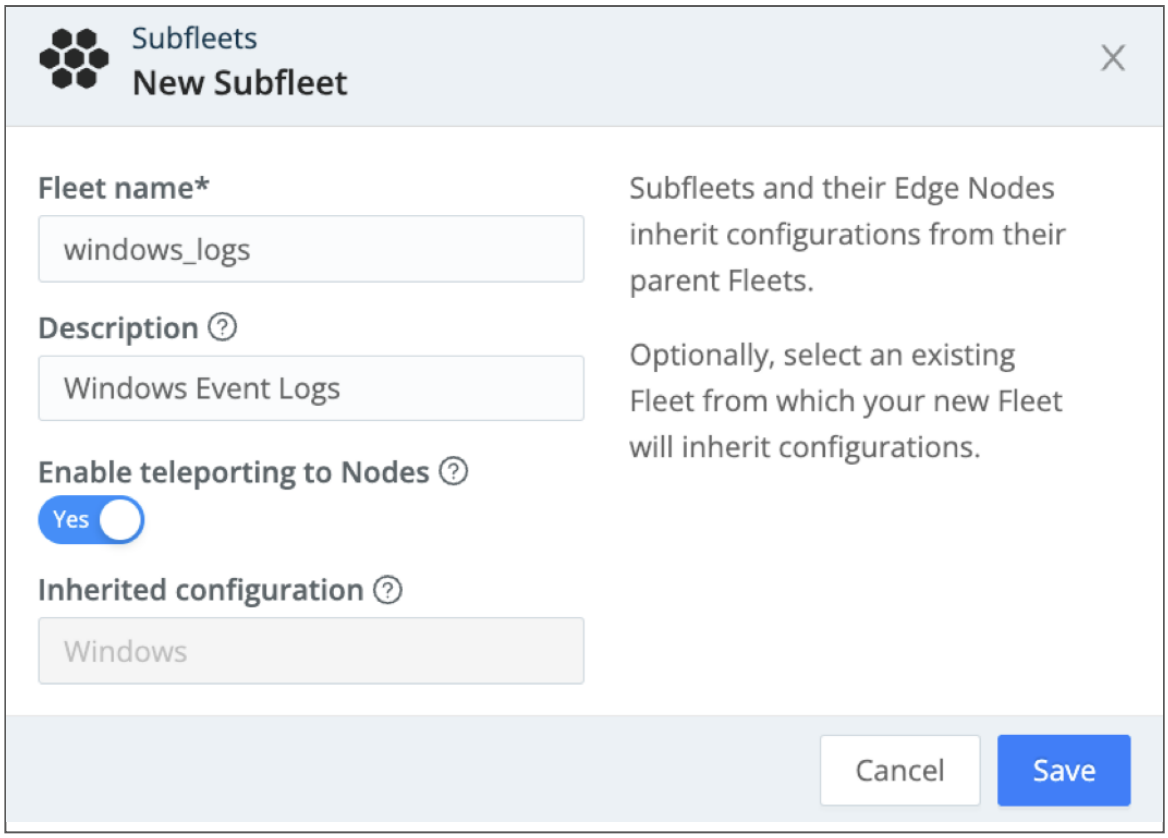
This section covers creating a Windows-specific Fleet, adding Windows Nodes via bootstrap scripts, and assigning Nodes to Fleets or Subfleets via mapping rules.

Create a Windows-Specific Fleet

Before we deploy the Edge Node, let's configure a Fleet. Fleets and their corresponding Subfleets allow you to author and manage configuration settings for a particular set of Edge Nodes. We recommend separating Linux and Windows machines into different Fleets per OS.

The first step is to configure a Fleet and name it: `Windows`.

For our mapping illustration later, let's also create a Subfleet in your `Windows` Fleet. In our example, we named it `windows_logs`.



Create a Subfleet

We use the name `Windows` for demonstration purposes and simplicity. Since Fleet names must be unique across your deployment, you should select a name that makes sense in practice. For better practices on designing and naming your Fleets, see the Fleet Hierarchy and Design doc.

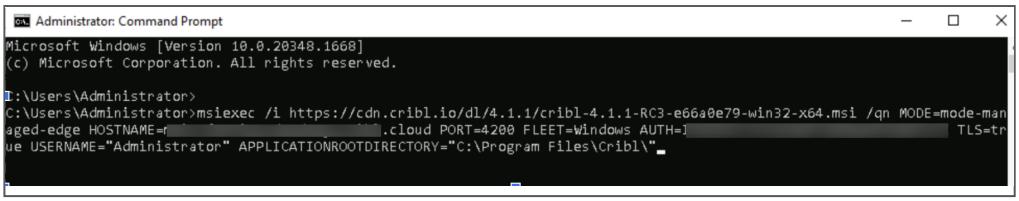
Add a Windows Edge Node Using the Bootstrap Script

Next, add a new Windows Edge Node in your Fleet with the built-in bootstrap script.

1. Navigate into the Windows Fleet you just configured.
2. On the Fleet page, select **Add/Update Edge Node** and select **Windows > Add**.
3. Copy the **Command Line** Script if you're using CMD or **Powershell** if you're using Powershell.
4. Paste the script into your command line shell and execute it.

You must run your choice of shell as a Windows Administrator for the bootstrap script to work. For details, see Windows resources:

- Start Command Prompt as an Administrator.
- Starting Windows Powershell with Administrative Privileges.



Pasting the script to the Command Prompt

Starting in Cribl Edge 4.1, we introduced the following changes for Windows deployments:

- The MSI installer uses the `CRIBL_VOLUME_DIR`. Cribl Edge stores logs, configs, state, and so on in `ProgramData\Cribl`.
- The default installation type is an Edge single-instance deployment.
- The bootstrap script includes TLS when you deploy from a Cribl.Cloud Leader.

Editing the `instance.yml` Files

You can edit the `instance.yml` files (located in `ProgramData\Cribl`) to change options like the following:

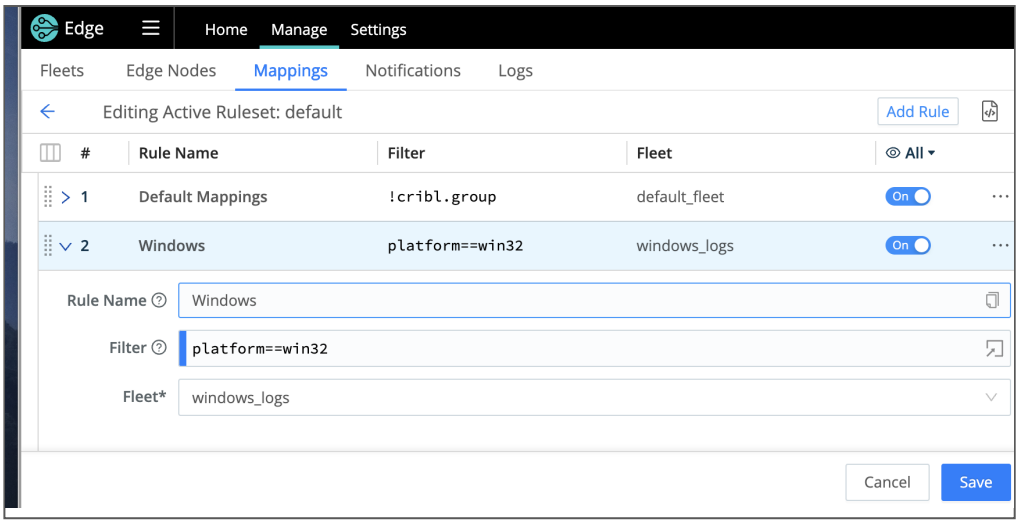
- Reassign the Edge Node to a different Fleet.
- Enable TLS after installation (for versions of Cribl Edge older than 4.1). For details, see Enabling TLS After Installation.

For details, see Using YAML Config File to configure an Edge Node.

Use a Mapping Ruleset to Switch Fleets

In this step, we will show you how to create a [Mapping Ruleset]mapping-edge-nodes to switch the Windows Edge Node to the Subfleet `windows_logs` you created earlier.

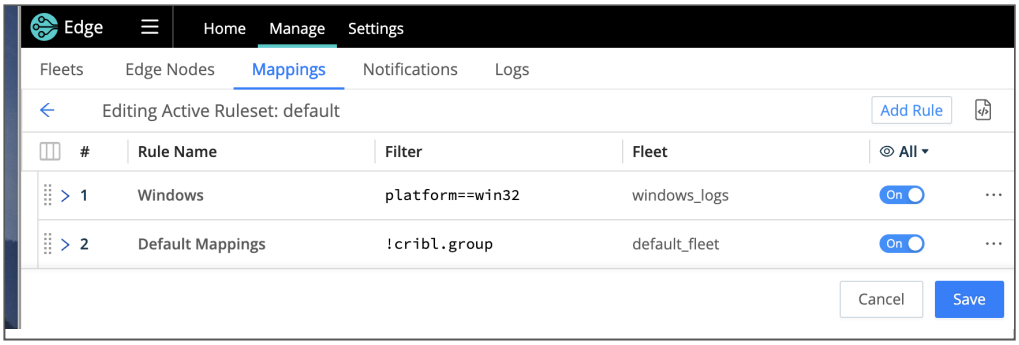
First, we'll add a new rule in the existing `default` ruleset.



Map to the SubFleet

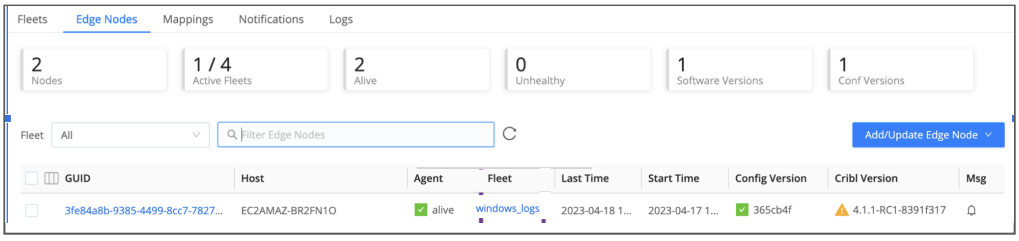
Only one Mapping Ruleset can be active at a time, although a ruleset can contain multiple rules. It's important to keep your Fleet assignments within one ruleset. This example uses the `default` ruleset.

Make sure you reorder the rules, so the new rule is first.



Reorder the Rulesets

Your Windows Edge Node will now be assigned to the new SubFleet.



View Edge Nodes

For details on mapping, see [Mapping Edge Nodes to Fleets]mapping-edge-nodes.

Explore Your Windows Node

When you first log in to Cribl Edge on Windows (single-instance or managed node), you'll land on the **Home** tab where you can explore the metrics and log data that the Node has auto-discovered and manually discover and explore other data of interest. For details, see Explore Cribl Edge on Windows.

From the Fleet's **Explore** menu, select the **System State** tab to see snapshots of the host system's current state in configurable time intervals. For details on what's displayed, see System State.

You can specify which collectors are enabled and the polling interval, by configuring the System State Source.

Configure the Windows Event Logs Source on Cribl Edge

Next, we'll look at how to get Windows Event Logs into Edge, and on to your chosen destination. We must first identify the logs we want, and then configure Edge's Source to match.

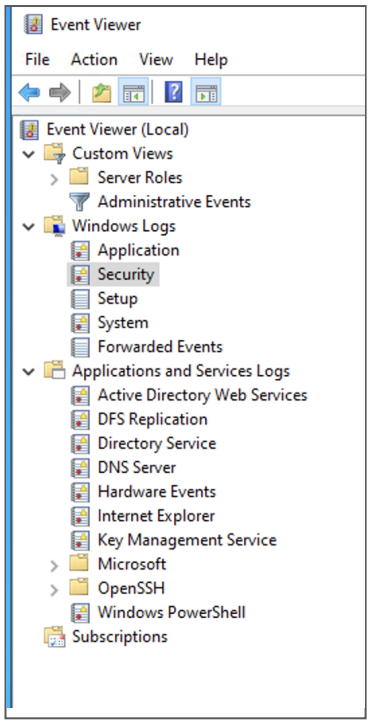
Locate Windows Event Logs in the Server

In order to configure the Windows Event Logs Source on Cribl Edge, you need to figure out which event logs you want to collect from your Windows Server.

Event logs in Windows Servers are classified into the following broad categories:

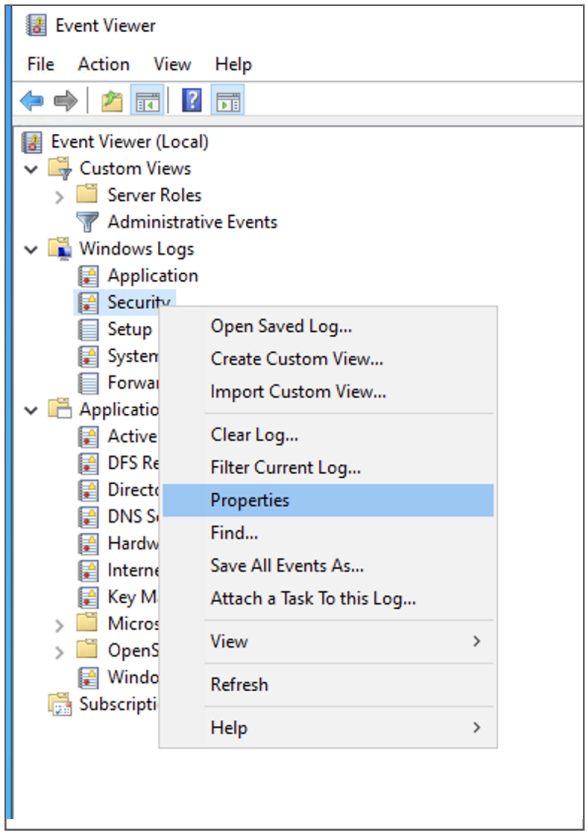
- **System**: Events related to the system and its components. Failure to load the boot-start driver is an example of a system-level event.
- **Applications and Services**: Events related to software or an application hosted on a Windows computer get logged under the application event log. For example, 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, go to your Windows Server Event Viewer.



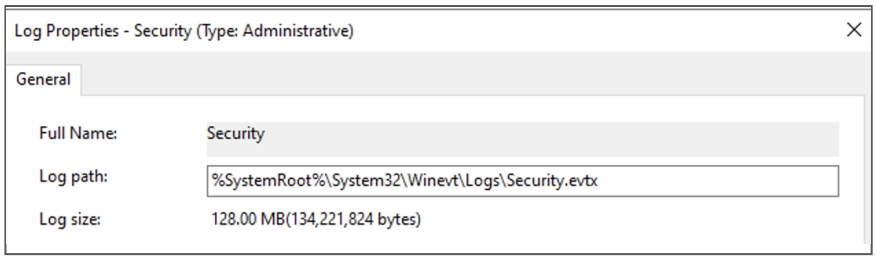
Event Viewer

To get the accurate name of the event log, right-click and select **Properties**.



Log properties

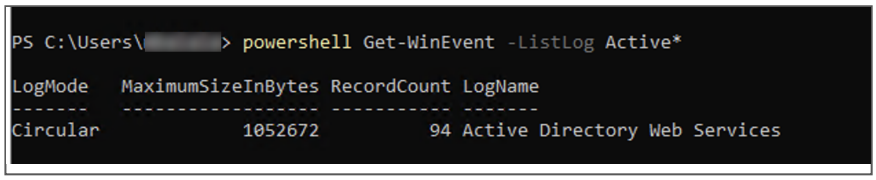
From the **Properties** modal, copy the **Full Name** value to your clipboard. In the next section, when you populate the Edge Windows Event Log Source's configuration modal, you'll paste this into that modal's **Event Logs** field.



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*
```

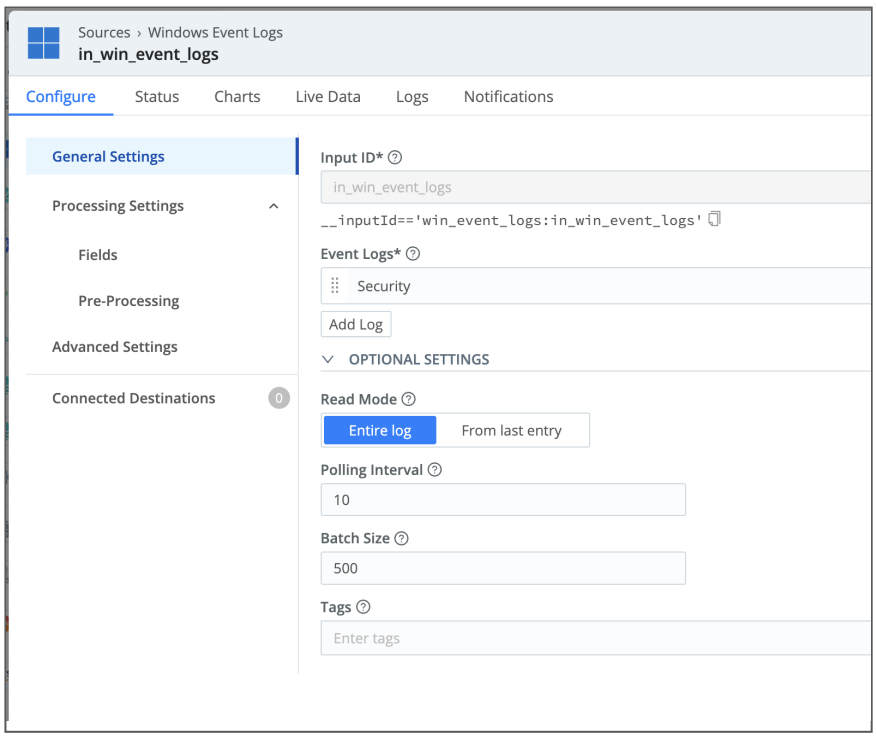


Log Name from PowerShell

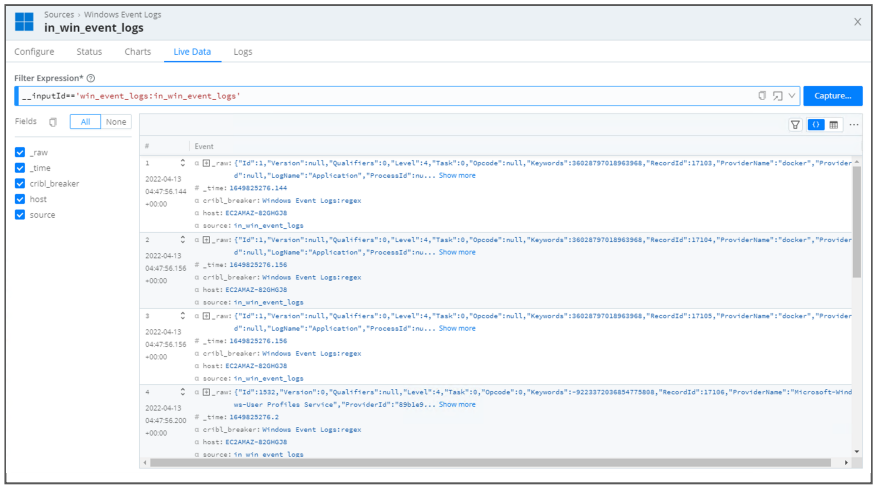
From the powershell output, the **LogName** field is what you would specify in the Cribl UI. In the above screenshot, for example, you would enter `Active Directory Web Services` in the Cribl UI.

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.



When done, **Commit** and **Deploy** your changes. Before moving on to the next step, check the **Live Data** tab to make sure the logs are generated.



Live Data

You can now send the event logs to your visualization tool of choice or any of Cribl Edge's supported Destinations.

Configure the Windows Metrics Source on Cribl Edge

To collect Windows metrics rather than log events, configure and enable Cribl Edge's Windows Metrics Source.

The key requirement here is to decide on the level of granularity for the metrics you want to collect. In the **Windows Metrics** Source, go to the **Host Metrics** left tab and select its **Custom** button. Here, we recommend setting the following granularity options for three of the configurable types:

- **CPU**: Enable **Per CPU** metrics.
- **Network**: Enable **Per Interface** metrics.
- **Disk**: Enable **Per Volume** metrics.

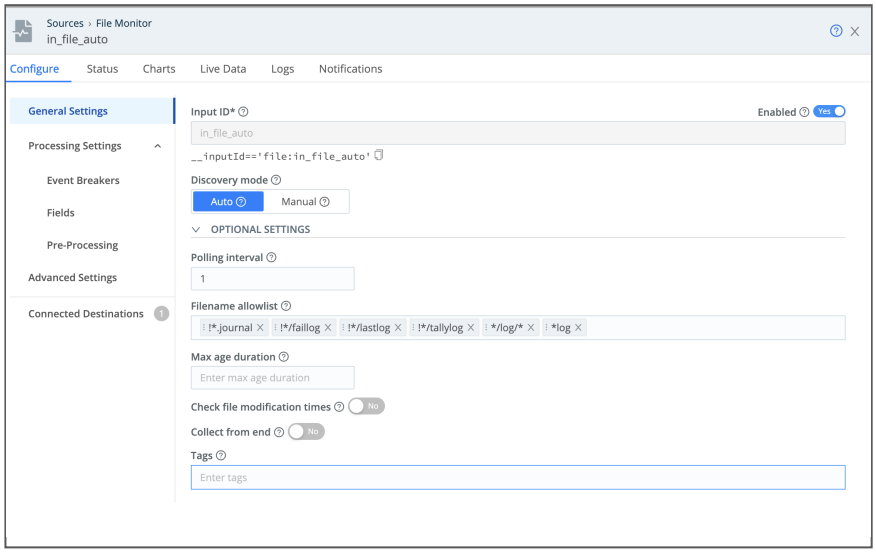
On the **Pre-Processing** tab, select the `prometheus_metrics` Pipeline. This will format the metrics for export to Prometheus-compatible services like Grafana.

When done, **Commit** and **Deploy** your changes. Before moving on to the next step, check the **Live Data** tab to make sure the metrics are generated.

To see an example of how to connect to your Windows Metrics Source to a Grafana Cloud Destination and visualize the data in Grafana, see Monitoring your Infrastructure with Cribl Edge.

Configure the File Monitor Source on Cribl Edge

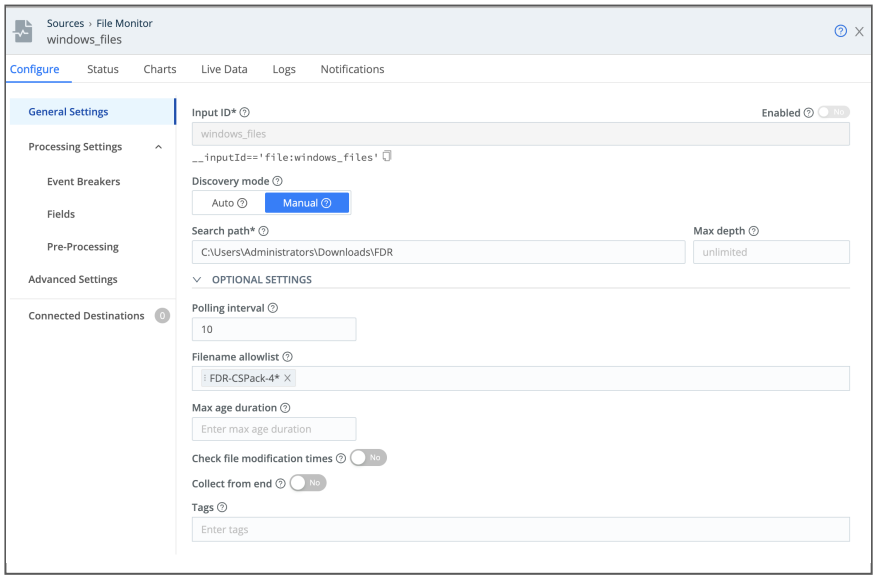
To round out this walk-through, we'll take a quick look at enabling the File Monitor Source to gather application logs. The example configuration below monitors all files that end with variations on `.log`, as well as `.journal` files.



File Monitor in Auto mode

The Leader deploys with two File Monitor Sources preconfigured, respectively to operate in **Auto** and **Manual** modes. **Manual** mode enables you to specify a single path from which Edge will collect log files. You define this path (with no wildcards) in the **Search path** field shown below.

Manual mode also provides a **Max depth** setting where you can limit collection 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.



Update the **Filename allowlist** with the specific file to monitor; this field supports wildcards. For guidance on how to use allowlists without creating duplicates, see Using Allowlists Effectively.

Bonus Section: Executing PowerShell Commands

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 Sources.

The Exec Source command is actually executed from a Windows command prompt, not from a PowerShell prompt.

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.

The main thing to keep in mind when you configure Cribl's Exec Source is that what you enter in the **Command** field is slightly different from a PowerShell prompt command.

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 as follows:

To process the PowerShell command for this script: `Get-CimInstance win32_process | select ProcessId,Name, commandline,HandleCount,WorkingSetSize,VirtualSize`

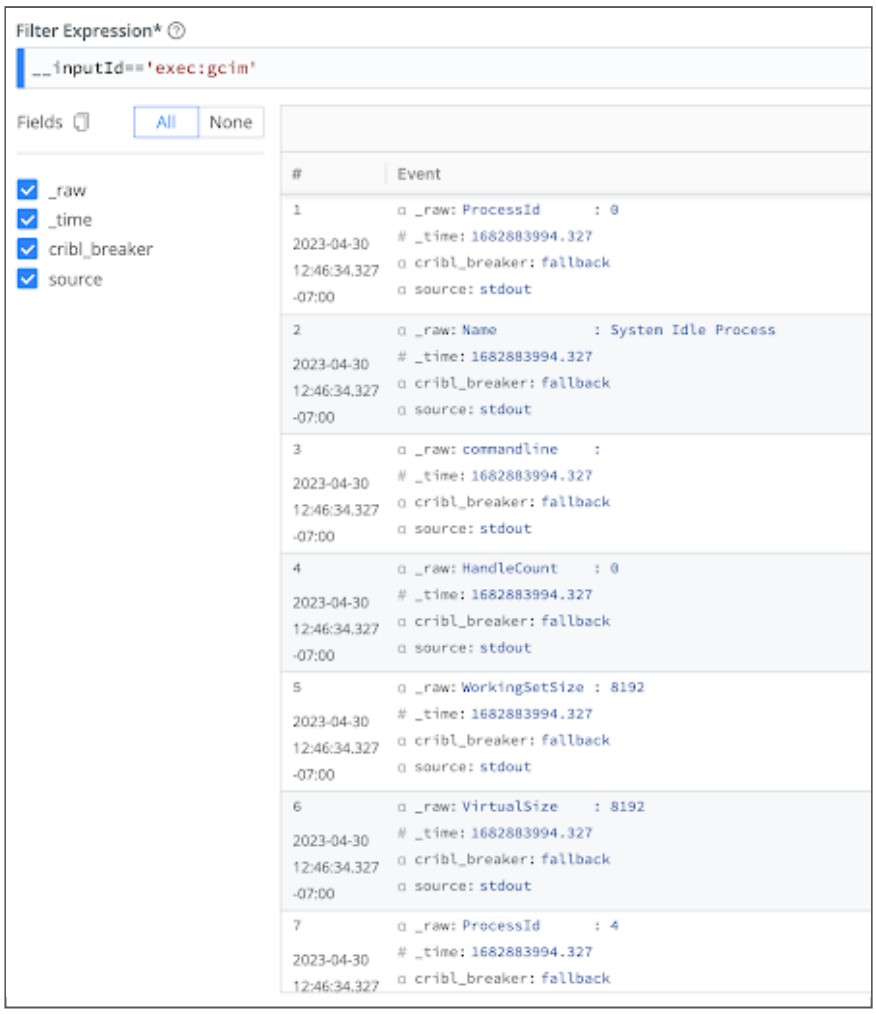
...enter the following into the Exec Source's **Command** field:

```
powershell [-command] "Get-CimInstance win32_process | select ProcessId, Name, commandline, HandleCount, WorkingSetSize,VirtualSize"
```

(Include the `-command` parameter if your PowerShell version requires it.)

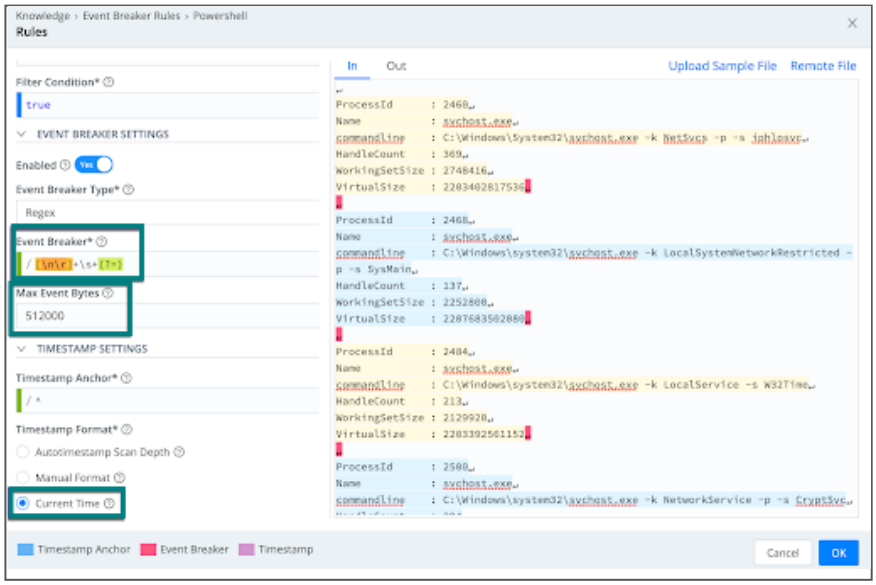
Exec Source's Command

The output of the command without using an Event Breaker:



Default Output

The output of the command with an Event Breaker:



5.6. Monitor And Troubleshoot Cribl Edge

Cribl Edge is a data collection agent for Windows, Linux, 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.
- **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 a centralized monitoring 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

The **System Activity** tab in the **Node Info** modal provides a detailed view of your Edge Node's health, including CPU usage, memory consumption, network traffic, and disk operations. This information helps you identify potential performance bottlenecks and ensure optimal data processing. To view **System Activity**, click the row for an Edge Node. For details, see Viewing the System Activity for a Node.

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

Analyze logs regularly to identify trends and potential issues.To balance detailed logging with performance, adjust the log level for each logging channel on the Cribl Edge Settings page. You can also customize the redaction of sensitive fields.

We recommend setting the following to `debug` (which is the most verbose level):

- `cribl.log`
- `CriblMaster`
- `Edge`

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.
- Monitor the Edge's resource utilization to identify any bottlenecks. For details, see Viewing the System Activity for a Node.

Refer to Common Challenges on the Edge topic for other troubleshooting tips.

6. 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.

A centralized interface allows you to manage all your Workspaces and control access for Members and Teams to individual Workspaces.

Workspaces require Cribl.Cloud with certain plan tiers. 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.

Workspaces require Cribl.Cloud with certain plan tiers. For details, see Pricing.

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.

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.

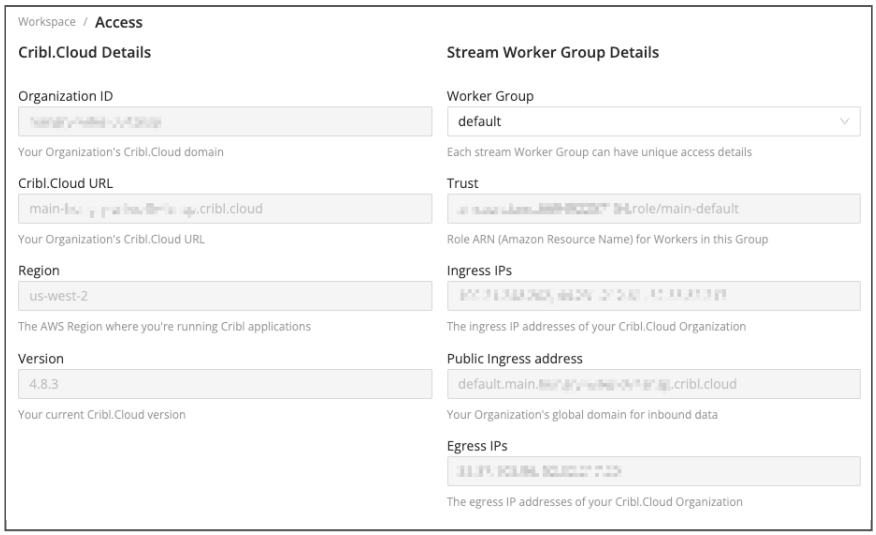
For a provisioned Worker Group, you see the following information:

Field	Description
Trust	Role ARN for Workers in this Worker Group. You configure these ARNs on the Trust tab.
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.
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.

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 certain plan tiers. 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 ARN (Amazon Resource Name) for your Workspace, in the sidebar, select **Workspace** and then **Trust**.

You can copy and paste the **Worker ARN**s listed in the **Trust** page to attach a Trust Relationship to an AWS account's IAM role. Use the **Group** drop-down to display the ARN for any Group of Cribl-managed Stream Worker Groups in Cribl.Cloud.

Attaching a 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.

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. Cribl.Cloud will display a banner, notifying you that rules edits are 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.

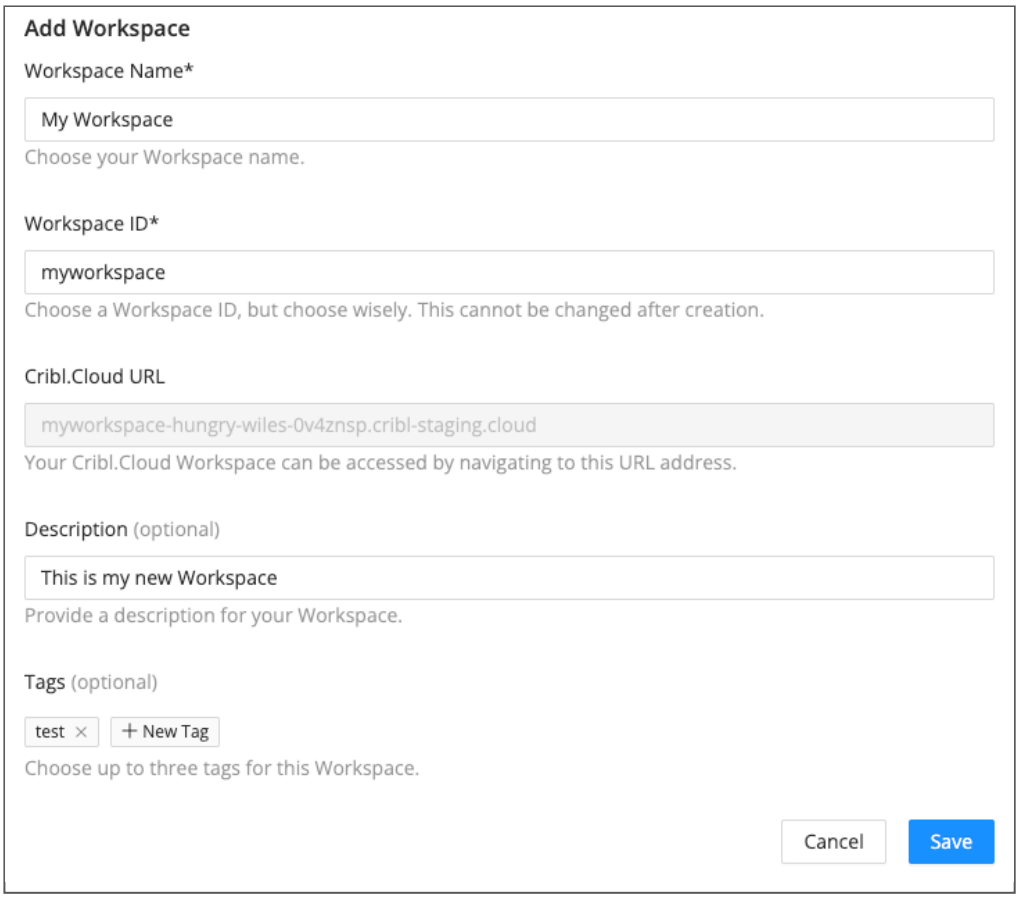
6.1. Configuring Workspaces

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



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.
3. Here you can change the name, description, and tags of a Workspace. However, you can't change the Workspace ID after it has been created.
4. Confirm with **Save**.

On the same page you can select **Delete Workspace** to remove it. You will need to manually type the name of the Workspace as a security measure.

Only Members with the Organization Owner permission can delete Workspaces.

7. Administering

7.1. Upgrading Edge

7.1.1. Upgrading Overview

Learn about the ways you can upgrade Cribl Edge, as well as general and version-specific upgrade considerations.

Ways to Upgrade Cribl Edge

You can upgrade Cribl Edge through these methods:

- Upgrade from the UI to upgrade the Leader and set a target version for each Fleet.
- Upgrade Manually using a command-line interface to upgrade Edge by uncompressing a new version on top of the old one.
- Upgrade Containers running Cribl Edge using Docker or Kubernetes.

Upgrading Requirements and Caveats

Here are some considerations to review before you upgrade.

- Upgrading Edge Nodes from the Leader is available on certain plan/license tiers. For details, see Pricing.
- 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.
- Review Edge and Leader Nodes Compatiblity.
- Before upgrading your Leader to v.4.0 and later, see Persisting Socket Connections to prepare the host to keep communications open from Edge Nodes.
- Cribl Edge 4.1 and later encrypts TLS certificate private key files when you add or modify them. See instructions for backing up your keys from earlier versions.

Keep Reading

- Better Practices for Upgrading Cribl Edge Nodes
- Troubleshooting Cribl Edge Upgrades

7.1.2. 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 Leader and Edge Nodes Compatibility.
- 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 Fleet Configuration 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.

Specific Considerations

In addition, review the considerations for each of the applicable scenarios below:

- Specific considerations for UI-based upgrades
- Specific considerations for upgrading containers
- Specific considerations for Edge Nodes deployed via RPM Packages
- Specific considerations for upgrading Windows Environments

Keep Reading

- Upgrading via UI
- Upgrading Manually
- Upgrading Containers
- Troubleshooting Cribl Edge Upgrades

7.1.3. Upgrading Via The UI

7.1.3.1. Upgrading Edge Leader Via UI

Transporting an Edge installation into the future.

This topic explains how to use the UI to upgrade a Cribl Edge Leader Node in a Linux distributed deployment.

If you need to upgrade a single-instance deployment of Cribl Edge, or you prefer to perform upgrade operations in a command-line interface, see the Upgrade Manually topic.

When you upgrade Cribl Edge via the UI, you must follow one of these upgrade paths:

Current Version	Upgrade Path
4.x	4.x
3.x	3.x through 4.x

Upgrade the Leader Node

To upgrade the Cribl Edge Leader, go to **Settings**, then select **Upgrade** under **Global Settings**.

Remember, these steps only apply to on-premises, distributed deployments of Cribl Edge, not Cribl.Cloud deployments or single-instance deployments.

You can choose to download upgrade packages from one of two sources:

- The Cribl content delivery network (CDN).
- A filesystem location that you specify in the form of a path. The running Leader application must have the connectivity and permissions to access the filesystem location.

Select a Package Source

Once you're in **Global Settings**, you will select between **CDN** and **Path** for your upgrade.

In Cribl.Cloud, **CDN** and **Path** options don't appear.

Package Source	Description	When You Select This Package Source
CDN	Downloads an installation package directly from Cribl's content delivery network.	You'll see the currently installed version and the version available on the CDN. The Leader will always upgrade to the current CDN version when you select **Upgrade**.
Path	You specify the path to the installation package.	You must specify paths to installation packages that have compatible versions and architectures with your Leader and Edge Fleets. For example, a Worker Group installed on an ARM64 architecture will require an ARM64 installation package.

Use the buttons to choose the desired package source. Then, the UI will display settings and information appropriate for the package source you chose, as described in the respective **CDN** and **Path** sections below.

Configure Settings for CDN Upgrade

If you select **CDN** as your upgrade method, you will see the following options:

- **Current CDN version**: Lists the latest version of Cribl Stream available on the CDN.
- **Leader**: This shows the currently installed version of Cribl Stream on the Leader. If a newer version is available, you will be able to use the **Upgrade to** button.
- **Stream Worker Groups**: Provides information and options for upgrading Stream Groups. See the Stream Documentation for more details.
- **Edge Fleets**: Displays **Upgrade Options**, including the deprecated **Enable Legacy Edge upgrades**.

Enable Legacy Edge Upgrades (Deprecated)

While this functionality still exists, it is deprecated and we do not recommend using it.

Defaults to `No`. Enabling Legacy upgrades removes the ability to automatically upgrade Nodes on a per-Fleet basis.

If you experience any upgrading issues with the 4.5.0+ Fleet upgrade functionality, consider toggling this to `Yes` to use the job-based upgrade framework and manually upgrade Fleets via the Fleet Upgrade UI and **Upgrade** button. You'll also need to toggle **Disable Jobs/Tasks** to `No`. See Disable Jobs/Tasks.

You may need to refresh the Global Settings page after enabling legacy upgrades to view the **Fleet Upgrade** tab.

Configure Edge Settings for Path Upgrade

With **Path** as your upgrade method, you'll see the following settings and information in the **Package Source** table:

- **Platform-Specific Package Location**: Enter or paste the path to the Cribl installation package. This can be either of the following:
  - An HTTP URL, for example `https://cdn.cribl.io/dl/4.1.0/cribl-4.1.0-6979aea9-linux-x64.tgz`
  - A local filesystem path, for example `myfolder/directory/cribl-package.tgz`
- **Package Hash Location**: Enter either of the following:
  - An HTTP URL, for example `https://cdn.cribl.io/dl/4.1.0/cribl-4.1.0-6979aea9-linux-x64.tgz.sha256`
  - A local filesystem path to the hash that validates the package.

  Supports SHA-256 and MD5 formats. You can simply append `.sha256` to the contents of the **Platform-specific package location** field.
- **Leader**: Indicates the version the Leader will be upgraded to when an upgrade is available and you select the **Upgrade** button.
- **Stream**: The version in this row will be used to upgrade the Stream Worker Group, because it matches the Leader's version.
- **Edge**: The version in this row will be used to upgrade Edge Fleets with a matching target version.
- **Version**: Displays the version of the upgrade package you added in this row.
- **Platform**: Displays the platform architecture for the package in this row.

Select **X** to immediately delete a row – there is no confirmation prompt.

You can add multiple rows to this table to specify packages for different versions and platforms/architectures. To obtain the latest packages from https://cribl.io/download, use the drop-down list to specify each platform (for example, x64 versus ARM). When you stage these packages on your own servers, preserve the original file names.

Leader area: Shows you the currently installed Leader version and whether an upgrade is available. Check the backup and rollback settings before you upgrade.

Stream Worker Groups: Provides information and options for upgrading Stream Groups. See the Stream Documentation for more details.

Edge Fleets: Offers **Upgrade Options** for Edge. See Enable Legacy Edge Upgrades for more information.

Missing Package Links

When you see this warning, it means there are Edge Fleets that can't be upgraded due to a missing package path.

To add a missing installation package:

1. Select the ? icon, which copies the path 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: `myfolder/directory/cribl-package.tgz`
4. Select **Save** to add the upgrade package path and resolve the warning.

7.1.3.2. Upgrading Edge Nodes Via UI

Learn how to upgrade Edge Nodes via the UI.

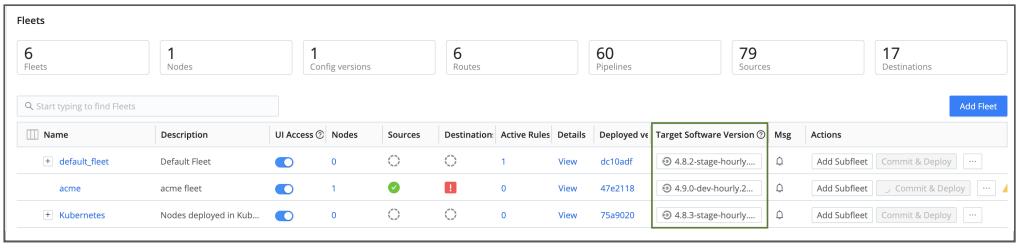
Upgrading Edge Nodes

You can upgrade the Nodes in a Fleet from the **Fleets** page. Upgrade options are on a per-Fleet basis:

- Choose **None** to never upgrade any of the Nodes in the Fleet.
- Select a target software version to upgrade all of the eligible Nodes in the Fleet to the selected version.

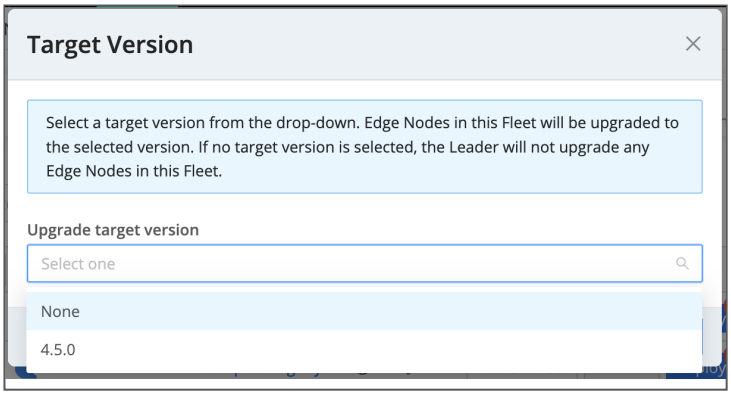
To upgrade the Nodes in a selected Fleet to a target version:

1. In the Leader Node's sidebar, select **Fleets**.
2. In the **Target Software Version** column, click the current version (or **None**, if applicable).



New Fleet upgrade interface

3. The Target Version window will open. Click the **Upgrade target version** drop-down and select a version.



Target version window

The next time the Node in the selected Fleet connects to the Leader, the Node will request the selected target version, and the Leader will upgrade the Node to that version.

Nodes in the selected Fleet will not upgrade to any future version until you select or change the target software version in the drop-down (even if you upgrade the Leader).

If you set the target software version to **None**, the Nodes in the selected Fleet will never upgrade. This lets you keep Edge Nodes at a specific version, even if you upgrade the Leader.

The **Add/Update Edge Node** command uses the **Target Software Version** for your Fleet to improve update accuracy. This applies to all supported environments: Docker, Kubernetes, Linux, and Windows. For details, see Edge Node Updates Using Fleet Version Settings.

Specific Considerations for UI-Based Upgrades

To choose the appropriate **Target Software Version** option consider the following scenarios.

- **None**: The Nodes in the Fleet will never be upgraded automatically. This option is suitable for highly critical Edge Nodes or Nodes managed by other departments that require strict control over upgrades.
- **Match Current Leader version**: The Nodes will be upgraded to match the software version of the currently running Leader Node. Once a Fleet is upgraded, it remains pinned to that specific version until:
    - The Leader itself is upgraded to a newer version.
    - An administrator manually intervenes and adjusts the target version for the fleet to the new leader version.

This option is useful for Fleets where you want to synchronize the Edge Nodes with the Leader's version to ensure compatibility and leverage the latest features.

Subfleets

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.

7.1.3.3. Upgrading Edge On Windows Via The UI

There are two ways to upgrade Edge Nodes installed on Windows, depending on the version of Edge you're running:

Cribl Edge Version	Upgrade Method
Version 4.3.1 and newer	Upgrade Windows Nodes from the Leader via the UI, or by re-running a Windows installer from the filesystem (manually upgrading).
Version 4.3.0 and older	Re-run a Windows installer from the filesystem. Upgrades, backups, and rollbacks, are not supported in the UI for Cribl Edge versions 4.3.0 and older. For details, see Manually Upgrading Windows Edge Nodes

Upgrading Windows Edge Nodes via the UI

Upgrade Windows Edge Nodes that are running version 4.3.1 and newer via the UI.

Specific Upgrade Considerations for Windows Environments

- **Endpoint Security Interference**: 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). See workaround: Common Challenges on the Edge | Cribl Docs.
- **ThreatLocker might block Microsoft Edge installation on Windows 11**: This causes silent failures. Contact your ThreatLocker admin to check for SYSTEM account blocks. They can create exceptions for Edge installations.

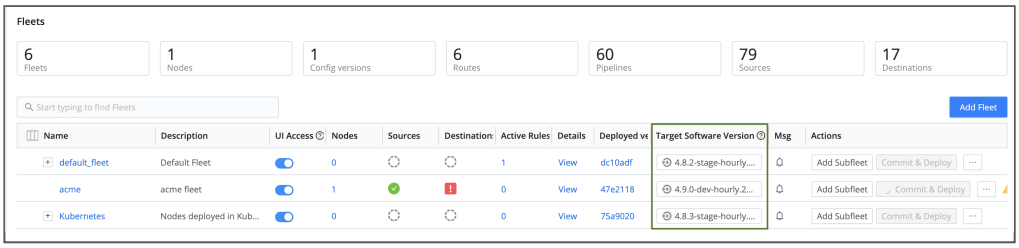
For details, see Debugging Cribl Edge on Windows.

7.1.3.4. Monitoring 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.



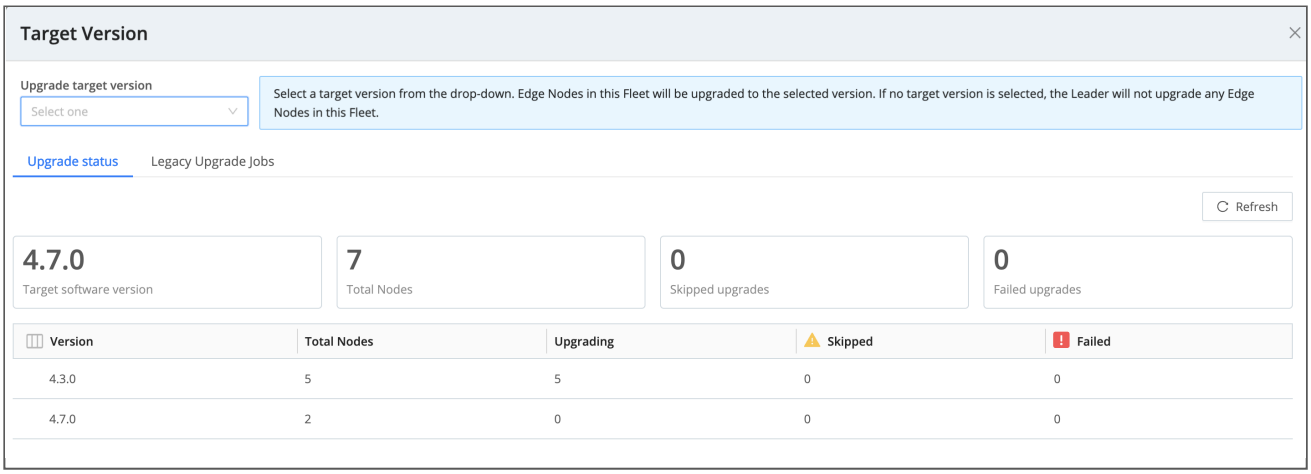
Target version button on **Manage > Fleets**

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
	  ◦ Version 2.4.3 and older for Linux 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.

7.1.3.5. 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 Cribl Edge to manage data on their edge nodes, divided into two distinct Fleets:

- **Fleet 1: Linux Operations.** This Fleet contains 20 Nodes, all running Edge v.4.4.4. When the IT admin upgrades the Leader of Fleet 1 to 4.5, 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, 4.4.4.
- **Fleet 2: Windows Monitoring.** This Fleet contains 15 Nodes. In this Fleet, the IT admin selects the current Leader version (in this case, 4.5) in the **Target upgrade version** drop-down. Now, when Nodes connect to the Leader, they pull down version 4.5 and upgrade to that version.

Fast forward four weeks: Cribl Edge has released a new version: 4.5.2. The Leader is now at version 4.5.2.

Since Fleet 1 does not have a target version selected, Nodes in Fleet 1 remain at their current version, 4.4.4. 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 4.5.2.

Nodes in Fleet 2 will remain on version 4.5, which was the Leader's previous version, until the admin selects the next **Target upgrade version** (4.5.2, 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).

7.1.3.6. Backup & Rollback Via The UI

Transporting an Edge installation into the future.

Backup and Rollback

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.

Edge can perform rollbacks only on Nodes that were running at least v.3.0.0 before the attempted upgrade.

Enable automatic rollback: Edge will automatically roll back an upgrade if the Edge server fails to start, or if the Edge Node fails to connect to the Leader. (Toggle to `No` 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`.

7.1.4. Upgrading Manually

7.1.4.1. Upgrading Manually Overview

Learn how to upgrade Edge Nodes manually using a command line interface (CLI).

In this guide, we'll cover benefits to upgrading manually and how to use the command line to upgrade your Edge Nodes.

A Word About Words

We will use "upgrading manually" and "on the command line / CLI" interchangeably in this article.

Why Upgrade Cribl Edge Manually?

With Edge, you have the option to upgrade manually or via the UI. Here are some use cases and benefits to using the command line to upgrade:

Flexibility and Control

Manually upgrading means you can 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.

Air-Gapped Environments and Custom Deployments

Consider upgrading via CLI when your organization can't connect to the internet for security or compliance reasons. You don't need to be online to upgrade on the command line.

When Cribl Edge is deployed in non-standard ways, such as via RPM package, within containers, or using a container orchestrator like Kubernetes, manual upgrades might be necessary.

Specific Upgrade Paths

When you upgrade Edge Nodes on a CLI, you don't have to worry about a version upgrade path - there are no restrictions on the versions you can upgrade to or from.

Keep Reading

- Upgrading Cribl Edge on Linux (CLI)
- Upgrading via RPM
- Manually Upgrading Cribl Edge on Windows
- Troubleshooting Cribl Edge Upgrades

7.1.4.2. Upgrading Cribl Edge On Linux (CLI)

Learn how to upgrade Edge Nodes manually using a command line interface (CLI).

Prior to upgrading, review Specific Considerations for Linux environments.

Upgrading a Linux Standalone/Single-Instance

This path requires upgrading only the single/standalone Node:

1. Stop Edge.
2. Download the package on your instance of choice here.
3. Uncompress the new version on top of the old one, e.g., in the `/opt/` directory:

   ```
   tar xvzf cribl-<version>-<build>-<arch>.tgz
   ```

   - On some Linux systems, `tar` might complain with: `cribl/bin/cribl: Cannot open: File exists`. In this case, please remove `cribl/bin/cribl` and untar again.
   - If you have **custom functions** in `cribl/bin/cribl`, please move them to `$CRIBL_HOME/local/cribl/functions/` before untarring again.
4. Restart Edge.

Upgrading a Distributed Deployment on Linux

For a Distributed Deployment, the general order of upgrade is:

1. Upgrade the Leader Node.
2. Upgrade the Edge Nodes.
3. Commit and deploy the changes on the Leader.

For distributed environments with a second Leader configured for failover, this is the upgrade order:

1. Stop both Leaders.
2. Upgrade the primary Leader.
3. Upgrade the second Leader.

4. Upgrade each Edge Node, respectively.

Upgrading the Leader Node

1. Commit and deploy your desired last version. (This version will be your most recent checkpoint.)
    - Optionally, `git push` to your configured remote repo.
2. Stop Cribl Edge.
    - Back up the entire `$CRIBL_HOME` directory (recommended, but optional).
    - Check that the Edge Nodes are still functioning as expected. In the absence of the Leader Node, they should continue to work with their last deployed configurations (optional).
3. Download the package on your instance of choice here.
4. Uncompress the new Edge version on top of the old one, e.g., in the `/opt/` directory:

```
tar xvzf cribl-<version>-<build>-<arch>.tgz
```

5. Restart Edge and log back in.
6. Wait for all the Edge Nodes to report to the Leader, and ensure that they are correctly reporting the last committed configuration version.

Cribl Edge's UI will not be available until the Edge version has been upgraded to match the version on the Leader. Errors will appear until the Edge Nodes are upgraded.

Upgrading the (Linux) Edge Nodes

These are the same basic steps as when upgrading a Single Instance, above:

1. Stop Cribl Edge on each Edge Node.
2. Download the package on your instance of choice here. If the Leader is on a prior release, see Cribl Past Releases for older packages.
3. Uncompress the new Edge version on top of the old one, e.g., in the `/opt/` directory:

```
tar xvzf cribl-<version>-<build>-<arch>.tgz
```

4. Restart Edge.

Commit and Deploy Changes from the Leader Node

1. Ensure that newly upgraded Edge Nodes report to the Leader with their new software version.
2. Commit and deploy the newly updated configuration **only after all** Edge Nodes have upgraded.

7.1.4.3. Upgrading RPM Installations Of Cribl Edge

Considerations for Upgrading RPM Installations of Cribl Edge

Cribl Edge Leader cannot directly upgrade Edge Nodes installed using RPM packages. Upgrading through the Leader might disrupt the RPM package manager.

Recommended method: Manual Upgrade with Package Manager.

Manually Upgrading RPM Installations of Cribl Edge

1. Download the new Cribl Edge RPM package for your desired version.
2. Use your system's package manager (e.g., `yum`, `apt-get`) to upgrade the existing Cribl Edge package. This approach offers more granular control but requires manual intervention on each node.

You must use the command line:

```
sudo yum update <CDN URL for new RPM>
```

We've disabled UI upgrades, because upgrading from the Leader bypasses the security of RPM.

Any changes you make to environment variables, like editing or adding new variables, are maintained when you upgrade, as well as when the service starts (on boot or manually).

For large organizations, it's recommended to establish 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.

7.1.4.4. Manually Upgrading Windows Edge Nodes

Upgrade Cribl Edge Nodes on Windows.

Manually Upgrading Windows Edge Nodes

These are the steps to upgrade Windows Edge Nodes that are on version 4.3.0 and older.

Test this upgrade on a staging environment before making these changes to your production Fleet.

To manage Windows Nodes from the Leader, upgrade to version 4.3.1 or newer.

Consider using existing out-of-band automation to perform the following steps:

1. Stop Cribl Edge on each Edge Node.
2. Uninstall the existing version of Cribl Edge:
    - If you have the original MSI that you installed, you can run:

      ```
      msiexec /x cribl.xxx.msi
      ```

    - If not, you can get the `GUID` from:

      ```
      HKEY_LOCAL_MACHINE > SOFTWARE > Microsoft > Windows > CurrentVersion > Unin
      ```

    And then run `msiexec /x {GUID}`

    - If you prefer to use PowerShell to uninstall the MSI, run:

      ```
      $app = Get-WmiObject -Class Win32_Product -Filter "Name = 'Cribl Edge'"
      $app.Uninstall()
      ```

3. Install the new Cribl Edge version:

   ```
   msiexec /i cribl-<X.X.X>-*-win32-x64.msi /qn /l*v C:\cribl-install.log KEEPDATA=
   ```

`KEEPDATA=1` persists data between installs. `KEEPDATA` will not overwrite the existing configurations, state, logs, etc.

When an older Edge Node is uninstalled, the product configurations are left behind under `C:\ProgramData\Cribl`, which the installer can reuse for the new Cribl Edge Node.

As soon as the new MSI is installed, it will automatically connect to the same Leader and join the same Fleet as before.

The Leader can upgrade Windows Nodes by running Cribl Edge 4.3.0 or later under the `Local System` account, instead of under the `Administrator` account.

7.1.4.5. Manual Rollback

Learn how to rollback back an 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: `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 Past Releases** page: https://cribl.io/download/past-releases/.
3. Locate the earlier 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/4.1.0/cribl-4.1.0-6979aea9-linux-arm64.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/4.1.0/cribl-4.1.0-6979aea9-linux-arm64.tgz | tar zxv`
7. Repeat the preceding steps to adjust all Nodes to a compatible version.

7.1.5. Upgrading 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 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 later inform the Leader that they are containerized Nodes.
A Fleet contains **Kubernetes / containerized** Nodes running version 4.4. The Fleet has a set **Target Software Version** of 4.5.0.	Yes	• Nodes on version 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 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
```

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
```

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
```

Cribl Edge supports both Docker and `containerd` runtimes.

7.1.6. Troubleshooting Edge Upgrades

In case of upgrading emergency, break glass.

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.

Leader and Edge Nodes Compatibility

Leaders on v.4.2.x are compatible with Edge Nodes on v.4.1.2, v.4.1.3, and later. Due to a security update, Edge Nodes running on v.4.0.4 cannot receive configurations from v.4.2.x Leaders.

Upgrading to v.3.5.4 or Later

- If you're upgrading to v.3.5.4 or later, all Edge Nodes will need to be on the same version as the Leader. Leaders running v.3.5.4 and later test whether Edge Nodes are running a compatible version before deploying configs that could break Nodes' data flow. The Leader will prompt you to upgrade these Nodes as needed. For details, see the Edge 3.5.4 release notes.
- Version 3.5.4 is also a compatibility breakpoint for the Cribl HTTP Source and Destination, and for the Cribl TCP Source and Destination. When running on Cribl Edge 3.5.4 and later, these two Sources can send data only to Workers running v.3.5.4 and later, and these two Destinations can receive data only from Nodes running v.3.5.4 and later. When running on Edge 3.5.3 and earlier, these four integrations can similarly interoperate only with Nodes running v.3.5.3 and earlier.

Safeguarding Unencrypted Private Keys for Rollback

Cribl Edge 4.1 and later encrypt TLS certificate private key files when you add or modify them.

Before upgrading from a pre-4.1 version, 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 pre-4.1 version, see Restoring Unencrypted Private Keys.

7.2. Notifications Overview

With Notifications, you can set up alerts that notify Cribl Edge admins about issues that require their immediate attention.

Types of Notifications

In Cribl Edge, you can configure Notifications about:

- Sources (including Cribl Stream Collectors) with high, low, or no data flow rates.
- Destinations that are experiencing backpressure, reporting errors, or approaching their persistent queue threshold.
- Pending expiration of a Cribl Edge license.

Notifications are also sent as events to Cribl Edge internal logs – both application-wide, and with a filtered view available on affected Sources and Destinations. Cribl Edge stores these application-wide logs in `notifications.log` on the Leader Node. The Leader Node is also responsible for sending all Notifications.

License Requirements

Notifications require certain plan/license tiers. Without an appropriate license, the configuration options described below will be hidden or disabled in Cribl Edge. For more information on license types and a feature comparison, see our Pricing page.

Notifications and Targets

A Notification target allows you to specify where Cribl Edge delivers a Notification. 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 without a target).

For details, see Notification Targets.

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.

Notifications and RBAC

Notifications work with Cribl Edge role-based access control. For users with non-administrative permissions, their assigned Roles and Policies determine the Worker Groups on which they can view Notification messages, configure Notifications, and configure targets.

Set up Notifications

To set up Notifications, you can create either the Notification or the target first — the order doesn't matter. In general, this is the process for creating a Notification:

- Create a Notification about a Source or Destination, or pending license expiration.
- Add or create a Notification target. See Targets Tab for more information. If you don't have a Notification target, the **Messages** icon will indicate you have a new Notification.
- Test the target to ensure it works.

Create a Notification about a Source or Destination

To create a Notification for a Source or Destination:

1. Navigate to your Fleet.
2. Select More and choose your Source or Destination.
3. In the **Notifications** column, select **Notifications**.

For specific information about configuring these options, see Source-State Notifications and Destination-State Notifications.

Create a Notification about License Expiration

To create a license-expiration Notification:

1. Navigate to **Settings** > **Global** > **Licensing**.
2. Select **Add Expiration Notification**.

License-expiration Notifications are only available for customer-managed instances of Cribl Edge. For specific information about configuring these options, see License-Expiration Notifications.

General

ID: Provide a unique ID for the Notification in this section. Cribl Edge enables the Notification by default, but you can disable the Notification by setting **Enabled** to `No`.

Configuration

Define the triggering condition for the Notification in this section. Conditions vary by Notification type, so consult one of the following topics for the relevant details:

- Source-State Notifications
- Destination-State Notifications
- License-Expiration Notifications

Select **Add Target** to add a preexisting target. Select **Create Target** to create a new target.

If you select or add an email Notification target, an additional set of configuration options appear. See Configuring Email Notifications for more information.

Metadata

Metadata fields are user-defined fields included in the Notification payload. All Notification types can contain metadata.

Select **Add field** here to add custom metadata fields to your Notifications in the form of key-value pairs:

- **Name**: Enter a name for this custom field.
- **Value**: Enter a JavaScript expression that defines this field's value, enclosed in quotes or backticks. (Can evaluate to a constant.)

Once you've saved your Notifications, you can see Notification events specific to this Destination on the **Events** tab within the Destination configuration window. When you set Source-state Notifications, a corresponding **Events** tab is available on Source and Collector config modals. For a comprehensive view of all Notification events, see the system-wide Events Tab.

Manage Notifications

You can manage existing Notifications and targets by selecting **Notifications** in the sidebar. You can also reach this page by selecting an existing Notification in a Source or Destination modal or in **Settings** > **Global** > **Licensing**.

Notifications Tab

This tab lists all your configured Source-state and Unhealthy Destination Notifications, across all integrations, along with any configured license-expiration Notifications. You can't create new Notifications here, but you can disable or delete existing Notifications. For any individual Notification, select **View Events** to see events that have triggered the Notification.

To modify a Notification, select anywhere on its row.

Targets Tab

This tab is where you centrally configure and manage targets that are available across Cribl Edge – for all Sources, Destinations, and license-based Notifications. See Notification Targets for details.

To create a new target, select **Add Target**. To delete a target, select **Delete** in the appropriate row.

7.2.1. Source-State Notifications

In Cribl Edge, you can configure Notifications on Sources and Collectors to trigger under these conditions:

- High Data Volume
- Low Data Volume
- No Data

Read on for details about these conditions and how to configure appropriate Notifications.

High Data Volume

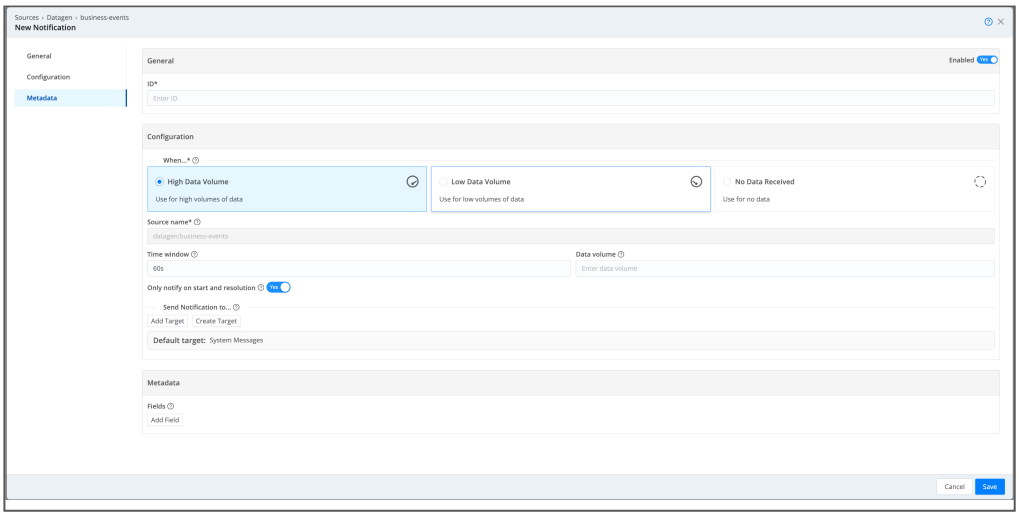
Cribl Edge will generate a Notification when incoming data over your configured **Time window** exceeds your configured **Data volume** threshold. This selection exposes the following fields:

Notification targets: The **Default target** is always locked to `System Messages`.

Source name: This field is locked to the Source on which you're setting 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 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.

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.



Configuring a high data volume Notification

Low Data Volume

Select the `Low Data Volume` tile to trigger Notifications when incoming data over your configured **Time window** is lower than your configured **Data volume** threshold.

This selection exposes the same additional fields as High Data Volume, except that here, the **Data volume** value defines a **floor** below which the Notification will trigger.

No Data

Select the `No Data Received` tile to trigger Notifications when the Source or Collector ingests zero data over your configured **Time window**.

This selection exposes the same additional fields as High Data Volume, except it omits the **Data volume** field. With no data entering the Source, there is no threshold to configure.

Source-State Notifications for Webhook Targets

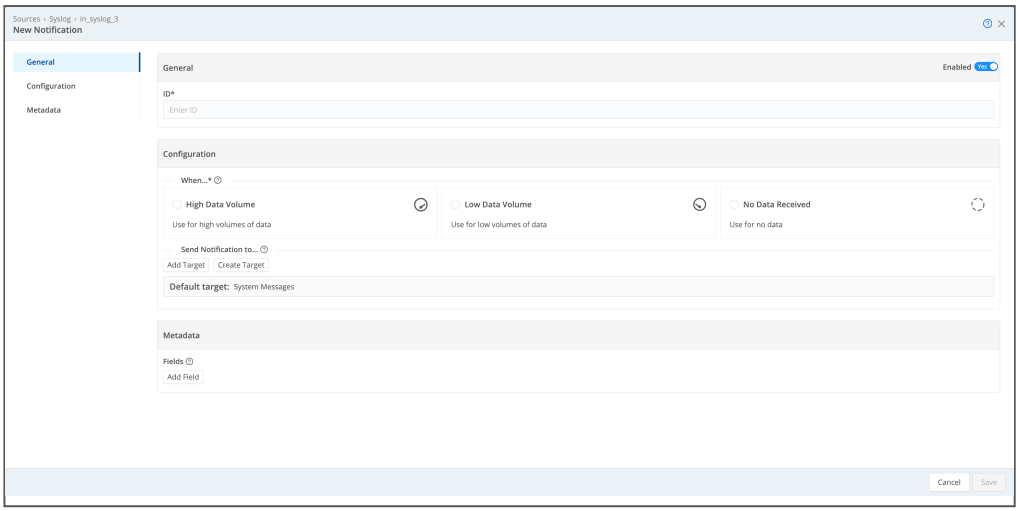
If you are sending a Source-state Notification to a Webhook Notification target, you can include a variety of expression fields in the target's **Source expression**. For more information, see:

- Fields Common to All Notification Types
- Source High Data Volume
- Source Low Data Volume
- Source No Data Received

Configure Source Notifications

To configure a Source-state Notification:

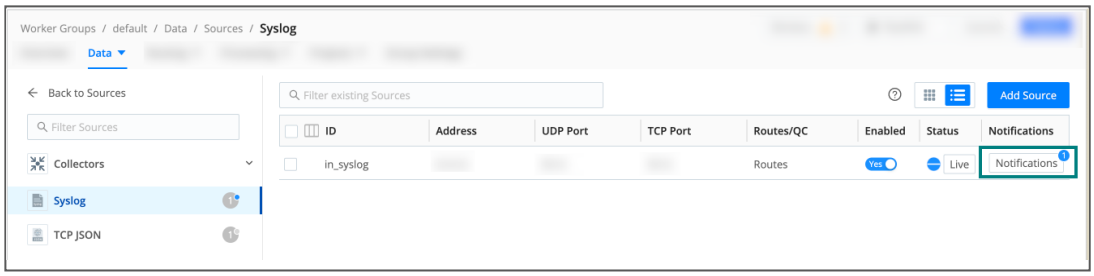
1. Configure and save the Source.
2. Access this Source's **Notifications** tab by one of the following methods:
    - Select the **Notifications** button on the Source page's appropriate row.
    - Reopen the Source's configuration modal and select its **Notifications** tab.
3. Select **Add Notification** to access the **New Notification** modal shown here.



Configuring a Source Notification

View Count of Configured Source Notifications

In the **List** view of a Source, you can quickly see how many Notifications are configured. The **Notifications** button will display a badge showing you the number of configured Notifications for the Source.



List view of a configured Source with Notifications

General

ID: Enter a unique ID for this Notification. Notifications are enabled by default, but you can disable the Notification by setting **Enabled** to `No`.

Configuration

When: Select one of the following Notification tiles:

- **High Data Volume**
- **Low Data Volume**
- **No Data Received**

You can create multiple Notifications for the same Source, but you must configure them separately.

Send notification to: Select **Add Target** to send this Notification to additional targets. You can add multiple targets.

- Use the resulting **Notification targets** drop-down to select any target you've already configured.
- Select **Create Target** to configure a new target.

See Notification Targets for details.

Default target is always locked to `System Messages`.

Source name: This field is locked to the Source on which you're setting 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 a Destination or 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.

Only notify on start and resolution: When this option is set to `Yes`, Cribl Edge will send a Notification at the onset of the triggering condition. On resolution, it will send a second Notification.

If you don't enable this option and a Source-state Notification's trigger condition persists beyond your configured **Time window**, Cribl Edge will send a new Notification, once per **Time window** interval.

Metadata

You can enter user-defined fields called *metadata*, which Cribl Edge includes in the notification payload. See Metadata for more information.

7.2.2. Destination-State Notifications

In Cribl Edge, you can configure Notifications on Destinations. You will receive a Notification to the set target when one of these conditions triggers:

- Backpressure activates on the Destination
- Persistent Queue Usage surpasses a set threshold
- The Destination is unhealthy

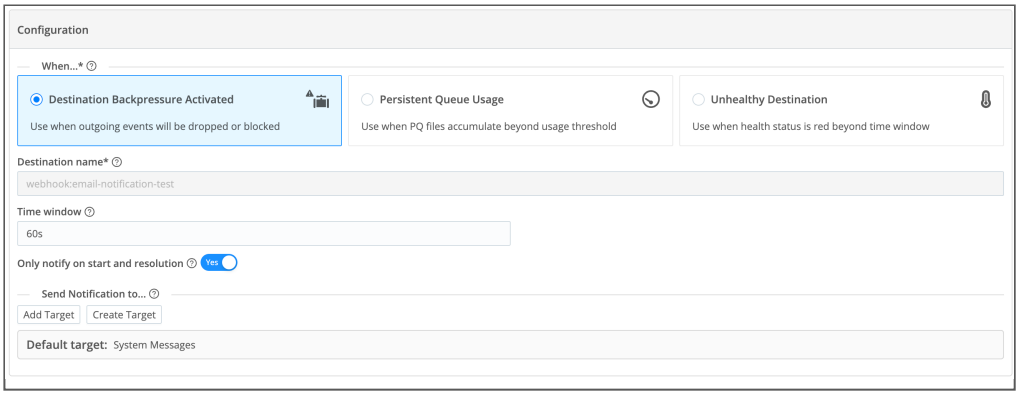
Read on for details about these conditions and how to configure appropriate Notifications.

Destination Backpressure Activated

Cribl Edge will generate a Notification when one of the following events occurs:

- 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, but its **Queue-full behavior** is set to either `Block` or `Drop new data`, and a filled queue causes the Destination to to block or drop outgoing events.

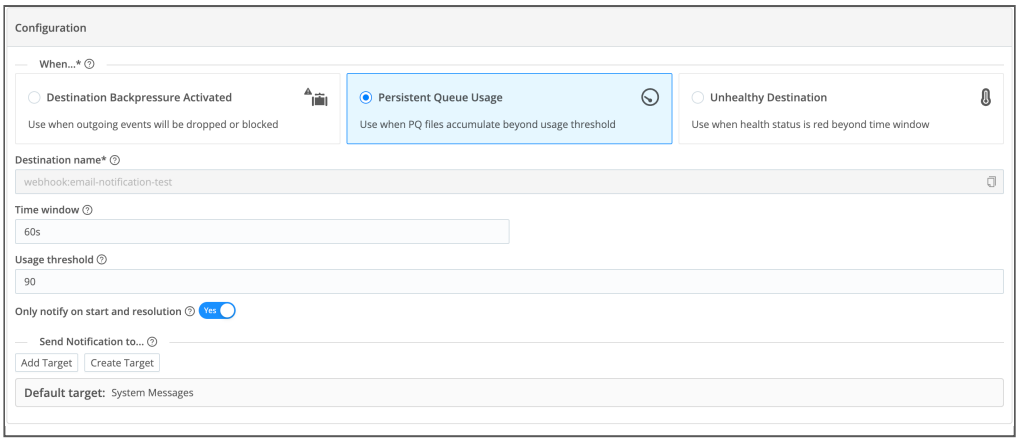
The threshold for the Notification to trigger is: Cribl Edge detected a blocked or dropped state during ≥ 5% of the trailing **Time window** that you configure in the Configuring Destination Notifications.



Backpressure Notification

Persistent Queue Usage

Cribl Edge will generate a `Persistent Queue usage has surpassed <threshold>%` Notification when the PQ accumulates files past the `<threshold>` percentage of capacity that you set in the **Usage threshold** field. This field appears only when you configure a Notification for **Persistent Queue Usage**.



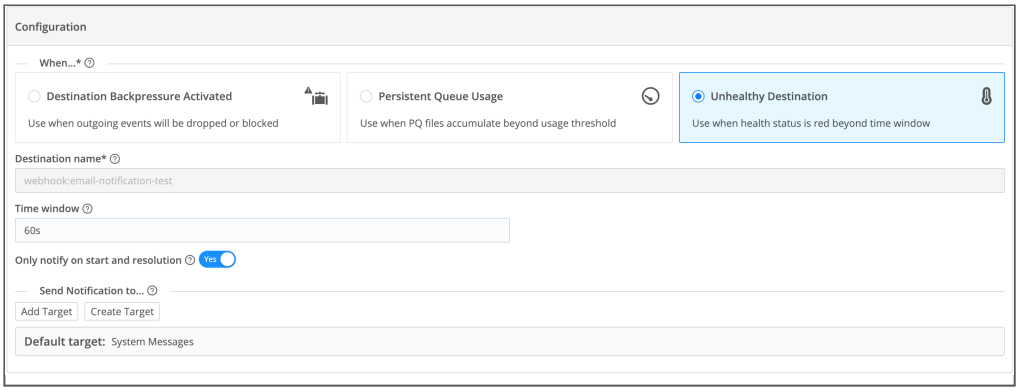
Persistent Queue Notification

Unhealthy Destination

Cribl Edge will generate a `Destination <name> is unhealthy` Notification when the Destination's health has been in red status (as indicated on the Monitoring page) over the trailing **Time window** that you configure in Configuring Destination Notifications.

The algorithm has slight variations among Destination types, but red status generally means that ≥ 5% of health checks, aggregated over the **Time window**, reported one of the following conditions:

- An error inhibiting the Destination's normal operation, such as a connection error.
- For multiple-output Destinations like Splunk Load Balanced or Output Router, > 50% of the Destination senders are in an error state.



Destination Unhealthy Notification

Destination-State Notifications for Webhook Targets

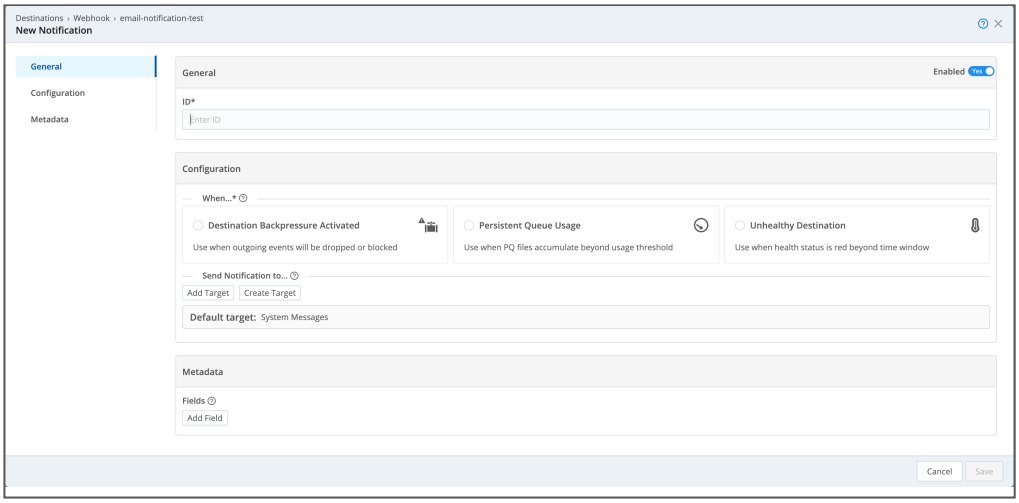
If you are sending a Destination-state Notification to a Webhook Notification target, you can include a variety of expression fields in the target's **Source expression**. For more information, see:

- Fields Common to All Notification Types
- Unhealthy Destination
- Destination Backpressure Activated

Configure Destination Notifications

To configure a Destination-state Notification:

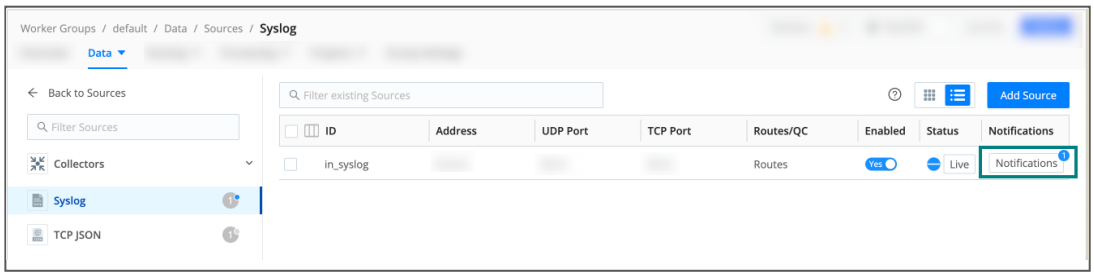
1. Configure and save the Destination.
2. Access this Destination's **Notifications** tab by one of the following methods:
    - Select the **Notifications** button on the Destination page's appropriate row.
    - Reopen the Destination's configuration modal and select its **Notifications** tab.
3. Select **Add Notification** to open the **New Notification** modal shown here.



Configuring a Destination Notification

View Count of Configured Destination Notifications

In the **List** view of a Destination, you can quickly see how many Notifications are configured. The **Notifications** button will display a badge showing you the number of configured Notifications for the Destination.



List view of a configured Source with Notifications

General

ID: Enter a unique ID for this Notification. Notifications are enabled by default, but you can disable the Notification by setting **Enabled** to `No`.

Configuration

When: Select one of the following Notification tiles:

- **Destination Backpressure Activated**
- **Persistent Queue Usage**
- **Unhealthy Destination**

You can set up multiple Notifications for the same Destination, but you must configure them separately.

Send Notification to: Select **Add Target** to send this Notification to additional targets. You can add multiple targets.

- Use the resulting **Notification targets** drop-down to select any target you've already configured.
- Select **Create Target** to configure a new target.

See Notification Targets for details.

Default target is always locked to `System Messages`.

Destination name: This field is locked to the Destination on which you're setting 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 a Destination or 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.

Only notify on start and resolution: When this option is set to `Yes`, Cribl Edge will send a Notification at the onset of the triggering condition and a second Notification to report its resolution.

If you don't enable this option and a Destination-state Notification's trigger condition persists beyond your configured Time window, Cribl Edge will send a new Notification, once per **Time window** interval.

Metadata

You can enter user-defined fields called *metadata*, which Cribl Edge includes in the notification payload. See Metadata for more information.

7.2.3. License-Expiration Notifications

To prevent interruptions in data throughput, you can configure a Notification that will be triggered two weeks before your Cribl Edge paid license expires, and then again upon expiration. (If the two-week Notification is cleared from the **Messages** tab between those dates, but the license has not been extended, it will trigger again.)

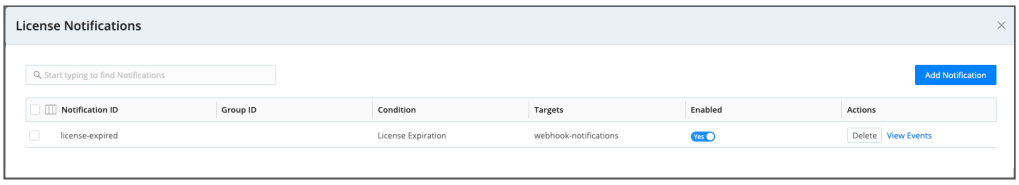
License-Expiration Notifications for Webhook Targets

If you are sending a license-expiration Notification to a Webhook Notification target, you can include a variety of expression fields in the target's **Source expression**. For more information, see:

- Fields Common to All Notification Types
- License Expiration

Configure License-Expiration Notifications

1. In the sidebar, select **Settings** > (**Global** >) **Licensing**.
2. Select **Add Expiration Notification** to access the **License Notifications** modal.



License Notifications modal

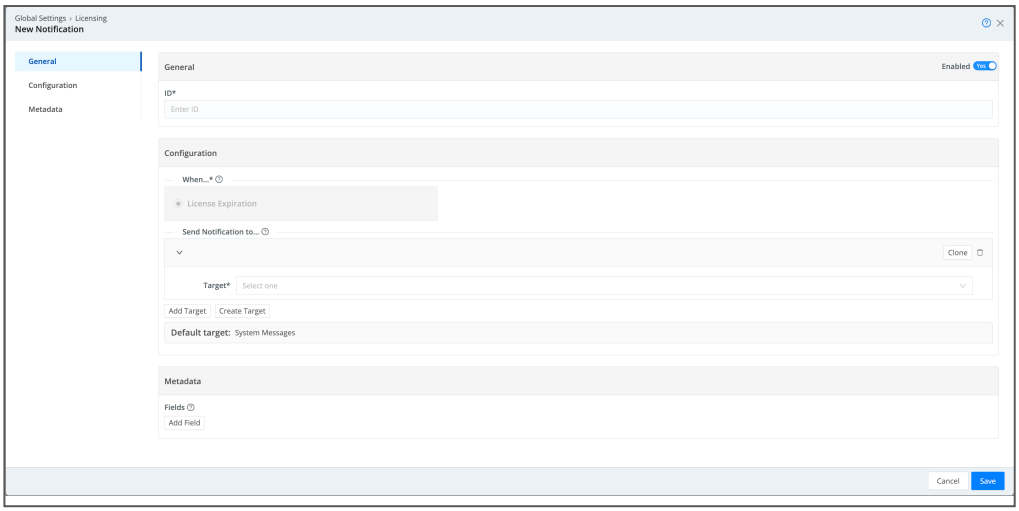
This modal shows existing license Notifications.

To modify a license-expiration Notification, select its row.

To delete a Notification, select **Delete**.

To view license-expiration events, select **View Events**.

Select **Add Notification** to to open the **New Notification** modal shown here.



Configuring a license expiration Notification

This **New Notification** modal provides General, Configuration, and Metadata tabs.

General

ID: Enter a unique ID for this Notification. Notifications are enabled by default, but you can disable the Notification by setting **Enabled** to `No`.

Configuration

When: This modal's triggering condition is locked to `License Expiration`.

Send notification to: This section contains a list of the targets receiving this Notification. The **Default target** is always locked to `System Messages`.

Select **Add Target** for each additional existing target that you want to send this Notification to. Select **Create Target** to create a new target for the Notification.

Metadata

You can enter user-defined fields called *metadata*, which Cribl Edge includes in the notification payload. See Metadata for more information.

7.2.4. 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.

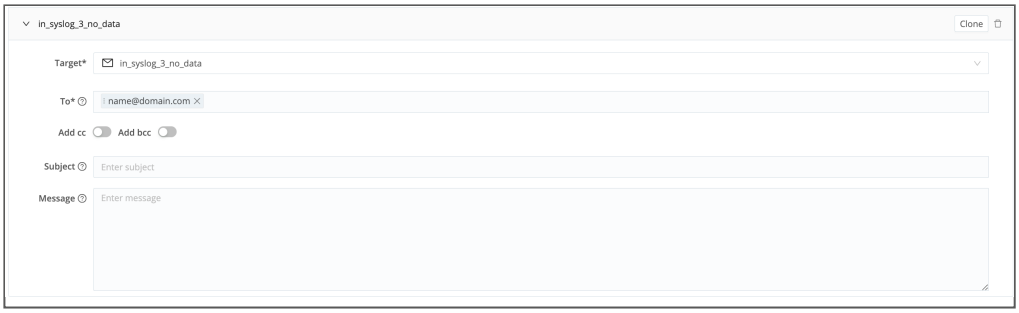
An email Notification requires two things — a configured Notification and an email Notification target. Cribl.Cloud Organizations using certain plan/license tiers also have access to a preconfigured email Notification target.

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	Example
`input`	Type and name of a Source or Collector.	`syslog:in_syslog_1`

Variable	Usage	Example
`output`	Type and name of a Destination.	`webhook:out_webhook_1`
`name`		Unique ID of a Source or Destination.
`bytes`	Quantity of data triggering the specified Notification.	`0`
`starttime`	Start time of an Event (in Epoch seconds).	`1706747185`
`endtime`	End time of an Event (in Epoch seconds).	`1706747294`
`health`	Health metric of the specified Source or Destination.	`0`
`_raw`	The `_raw` field of the Event triggering the Notification.	`_raw (Source ${name} in group ${__workerGroup} traffic volume greater than ${dataVolume} in ${timeWindow})`
`backpressure_type`	Backpressure behavior.	`1=BLOCKING, 2=DROPPING`
`queue_usage`	Percentage of capacity set in the **Usage threshold** field for a **Persistent Queue Usage** Notification.	`90`

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:

1. Select **Monitoring** > **Notifications**.
2. In the search field, search for **cribl_notification** fields that correspond to the Notification ID of the failed Notification.

7.2.5. Notification Targets

A Notification target specifies the delivery method for a Notification. Every Notification requires one or more targets.

Available target types include:

- Webhook
- PagerDuty integration
- Slack
- AWS SNS
- Email

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

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.

7.2.5.1. 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 to `Yes` 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, 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

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 `Yes`.

Round-robin DNS: Toggle to `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Edge to cycle through them in the order returned.

Compress: Compresses the payload body before sending. Defaults to `Yes` (recommended).

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`.

Max body size (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 **Max body size** 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 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.

7.2.5.2. 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).

7.2.5.3. 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 to `Yes` 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 to `No` (the 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 set to `Yes`, Cribl Edge to automatically retries HTTP requests that have timed out. Defaults to `No`. 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).

7.2.5.4. 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.

Region: Select the region associated with the Amazon S3 bucket.

Default Topic ARN: The default Amazon Resource Name (ARN) of the Amazon SNS topic to which you want to send Notifications. Cribl Edge expects the ARN in a format like this:

`arn:aws:sns:region:account-id:MyTopic`.

If you use a non-AWS URL, the format must be:

`{url}/myQueueName` – for example, `https://host:port/myQueueName`.

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.

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 to `Yes` 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. The default setting (`Yes`) 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 `Yes`.

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.

7.2.5.5. 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**.

If you are a Cribl.Cloud user on a certain plan/license tier, you can use the default email Notification target. This target requires no configuration.

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, orgs on a certain plan/license tier can use `system_email` – the default email notification target – 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 to `Yes` 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, and Search. 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@<workspace>-<org>.criblcloud.email`. To ensure delivery, recipients should add the `criblcloud.email` domain to their email allowlists.

7.3. Licensing

Every Cribl Edge download package comes with a Free license that allows for processing of up to 1 TB/day. This license requires sending anonymized Telemetry Data to Cribl.

Enterprise, Standard, and Sales Trial licenses are entitled to a defined, per-license daily ingestion quota. These licenses do not require sending telemetry metadata.

This page applies only to on-premises deployments. For basic information about Cribl.Cloud Enterprise and Standard plans, see Cloud Pricing.

Manage Licenses

Check Licenses

To check your license type, restrictions, and/or expiration date:

1. In the sidebar, select **Settings**.
2. On the **Global** tab, select **Licensing** to see these details.

Add or Renew Licenses

On the Leader (or in a Single-instance deployment), after submitting your newly untarred Cribl Edge software's registration page, you can add and manage licenses at **Settings** > **Global** > **Licensing**. Select **Add License** to paste in a license key provided to you by Cribl.

This applies to Cribl Edge Standard and Enterprise licenses, which must be renewed annually. Free licenses are already onboard the download package, need not be added or managed here, and do not expire.

Licenses in Airgapped Deployments

If your environment does not allow internet connectivity, you'll need to bypass the registration Web form when you deploy or upgrade Cribl Edge. To do so:

1. On the Leader's (or single instance's) host, navigate to `$CRIBL_HOME/local/cribl`.
2. Create a file named `license.yml`.
3. Paste in your license key in this format, then save the result:

```
licenses:
- <your-license-key-here>
```

Expiration/Renewal Notifications

In Cribl Edge 4.3 and later, the UI will prompt you to renew your paid license by displaying banner and drawer notifications, starting 30 days before license expiration. When you see these prompts, verify your license's expiration date and contact Cribl Sales at sales@cribl.io to renew.

Beyond these default notifications, you can configure custom License-Expiration Notifications. You can send these to PagerDuty, or to other external services via webhook.

License Quotas

Track Data Ingestion

To track your data ingestion volume over the last 30 days:

1. In the sidebar, select **Monitoring**.
2. Select **System** and then **Licensing**. This shows you the dashboard with your license consumption.

For higher-fidelity metrics, you can enable the Cribl Internal Source's CriblMetrics option to forward metrics to your metrics Destination of choice, and run a report on `cribl.total.in_bytes`.

Exemptions from License Quotas

Cribl does not require a separate license for sending data from Cribl Edge to Cribl Edge, such as sending from one Fleet managed by Leader Node A to a different Fleet managed by Leader Node B. In such situations, you can use the same license on Leader Node A and on Leader Node B.

- Data generated by Cribl Internal Sources normally does not count against your ingestion quota.
- Edge Node events that you drop in any Pipeline do not count against your ingestion quota.

If you are connecting Workers/Edge Nodes in a Cribl Stream Cloud hybrid deployment, you are granted additional exemptions to prevent double billing (in Cribl Edge, these exemptions also apply to customer-managed deployments):

- Data transferred between Cribl Stream Workers via the Cribl HTTP and Cribl TCP Sources does not count against your ingestion quota.

- Data that Cribl Edge sends to Cribl Stream from the Cribl HTTP and Cribl TCP Destinations counts against quota only in Cribl Edge.
- Data generated by Datagens does not count against your ingestion quota.

License Quotas with Compressed S3 Buckets

If you pull data from compressed S3 buckets, the license quota is applied to the uncompressed size of the file objects.

Enforce License Quotas

If your data throughput exceeds your license quota:

- Free and Standard licenses enforce data ingestion quotas through limits on the number of Worker Groups/Fleets and Worker/Edge Node Processes.
- Enterprise license keys turn off all enforcement.
- When an Enterprise or Standard license expires, Cribl Edge will attempt to fall back to a trial or free license, or – only if that fails – will block incoming data. For details, see Combining Licenses.

License Types

Cribl offers several Cribl Edge license types including Enterprise, Standard, and Free. For a current list of available license types and a detailed comparison of what's included in each, see Cribl Pricing.

Combine Licenses

Multiple licenses can coexist on one instance. When two or more Standard or Enterprise licenses are present, the effective ingestion quota for the deployment will be equal to the sum of their individual ingestion quotas.

When multiple *license types* coexist, the effective license is determined by the following priority order:

1. Enterprise or Standard licenses that have not expired.
2. Sales Trial licenses.
3. Free licenses.

License Expiration

When an Enterprise or Standard license expires, you can remove the old license to fall back to the Sales Trial or Free type. An expired Sales Trial license will fall back to a Free license, but you must remove the Sales Trial license and restart the Leader.

To delete your expired license:

1. In the sidebar, select **Settings**.
2. On the **Global** tab, select **Licensing**.
3. Locate your expired license, and select **Delete license**.
4. Restart the Leader.

Upon expiration of a paid license, if there is no fallback license, Cribl Edge will backpressure and block all incoming data.

Licensing in Distributed Deployments

With licenses that limit the number of Worker Processes/Edge Nodes, Cribl Edge will attempt to balance or rebalance Worker Processes (threads) as evenly as possible across all licensed Edge Nodes.

You configure licensing only on the Leader Node. (See Managing Licenses – Adding and Renewing.) The Leader will push license information down to Fleets as part of the heartbeat (Stream, Edge). There is no need to configure or store licenses directly on Edge Nodes.

Temporary License Restriction on Worker Node Startup

When you first start a Worker Node, it will use the default license. As a result, you might see a log entry warning about a limit on Worker Processes:

```
{
  "time": "2023-02-10T15:11:48.609Z",
  "cid": "api",
  "channel": "cribl",
  "level": "warn",
  "message": "number of worker processes has been limited by the license",
  "requested": 14,
  "licenseLimit": 10,
  "source": "cribl.log"
}
```

You can ignore this warning message. The license won't limit Worker Processes once the Leader passes the production license to the Worker.

Convert an On-Prem License to a Cribl.Cloud Billing Plan

You can convert on-prem Cribl Edge licenses to a Cribl.Cloud billing plan. This lets you keep your on-premises deployments of Cribl Edge, but manage all of the environment licenses in one interface.

For more details and setup steps, visit Connecting On-Premises Leaders to Cribl.Cloud Billing.

Telemetry Data

A **Free** license requires sharing of telemetry **metadata** with Cribl. Cribl uses this metadata to help us understand how to improve the product and prioritize new features.

Telemetry payloads are sent from all Cribl Edge nodes (Leader and Workers), to an endpoint located at `https://cdn.cribl.io/telemetry/`.

Test the Telemetry Endpoint's Connectivity

To manually test connectivity to the telemetry endpoint, especially if you are needing to configure a proxy, you can use the following command:

```
curl https://cdn.cribl.io/telemetry/
```

Expected response:

```
cribl /// living the stream!
```

If you get a 302 response code, check whether you've omitted the URL's trailing `/`.

Disable Telemetry

With an Enterprise or Standard license, you have the option to disable telemetry sharing from on-prem Cribl Edge. With a Free license, disabling telemetry will cause Cribl Edge to block inbound traffic within 24 hours.

If you would like an exception to disable telemetry in order to deploy in your environment, please contact Cribl Sales at sales@cribl.io, and we will work with you to issue licenses on a case-by-case basis.

Once you have received a license that removes the telemetry requirement, you can disable telemetry in Cribl Edge' at **Settings** > **Global** > **System** > **General Settings** > **Upgrade & Share Settings** > **Share telemetry with Cribl**. Set the toggle to `No`.

Metadata Shared Through Telemetry

Your Cribl Edge instance shares metadata with Cribl per interval (roughly, every minute).

Details sent for nodes in any mode:

- Version
- OS Distribution/Version
- Kernel Version
- Instance's GUID
- License ID
- Earliest, Latest Time
- Number of Events In and Out, overall and by Source type and Destination type
- Number of Bytes In and Out, overall and by Source type and Destination type
- Number of Open, Closed, Active Connections
- Number of Routes
- Number of Pipelines
- Runtime Environment Indicators (AWS and Kubernetes)

Additional details sent for Leader nodes (when Cribl isn't sending managed node data):

- Runtime Environment
- OS Distribution / Version
- Version
- Kernel Version

How We Use Telemetry Data

With telemetry enabled, Cribl software sends usage data directly to Cribl. We securely store and encrypt this data on Cribl-managed servers, with restricted access.

Cribl fully anonymizes and aggregates this usage data in our systems of analysis. There, we use the aggregated metrics to improve Cribl products and services by analyzing circumstances around disruptions, opportunities for ingest efficiencies, and ways to optimize performance. Access is restricted to only those Cribl employees and contractors who require anonymized data to perform their jobs.

Cribl collects License IDs only to ensure that all data is being sent by a Cribl product. These IDs cannot be used to personally identify any user of Cribl software or Cribl.Cloud services.

For further details, see the Cribl Privacy Policy.

Troubleshooting Resources

Cribl University offers a Troubleshooting Criblet on Switching from Free to Enterprise. 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 Troubleshooting Criblets and Advanced Troubleshooting short courses.

7.4. Uninstalling Cribl Edge From Linux

Removing a Cribl Edge installation, 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.

7.5. Uninstalling Cribl Edge 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
```

7.6. Fleet Settings

A Fleet's General Settings allows you to configure teleporting, throughput throttling, logging, upgrading, and security at the Fleet level.

To edit a Fleet's General 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 options on this tab:

Description: Optionally, add or edit a description of the Fleet's purpose.

Tags: Use tags to organize Fleets into logical categories. Then, you can search by tags on the **View all Groups** (Stream) or **View all Fleets** (Edge) interface. This search filters the list of Fleets, showing only those with the tag you entered.

Enable teleporting to Workers: Use this toggle to enable or disable authenticated access to Workers' UI from the Leader (Stream, Edge).

In Cribl Edge 4.1.2 and later, you can configure the following option in Shutdown Settings.

API Server Settings

General

You can set the following options for the API server.

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
- `0.0.0.0` for Leader, standalone Stream Workers, and Worker Groups.

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
- `9000` for Leader, standalone Stream Workers, and Worker Groups.
-

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

See Authentication Controls.

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

See Controlling Metrics Volume.

Storage

You can configure the following options for storage. In **General Settings**, open **Limits** and then select **Storage**:

Max sample size: 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`.

Max PQ size per Worker Process: Highest accepted value for the **Max queue size** 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

Disable jobs/tasks: Set to `Yes` by default in Edge Fleets and `No` by default in Stream Groups. In Edge, Nodes no longer poll the Leader for upgrade jobs. Setting this to `Yes` in Edge reduces application load from the Leader.

Nodes running 4.4.4 and older still upgrade via jobs and will honor the Jobs settings, even with the **Disable jobs/tasks** toggle enabled.

Job Limits

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 Group'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 Worker Nodes.

Task Limits

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`.

Max Task Usage Percentage: 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).x

Completion Limits

Artifact Reaper Period: Interval on which Cribl Edge attempts to reap jobs' stale disk artifacts. Defaults to `30m`.

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 Worker Node. Defaults to `500`. Minimum `0`.

Task Manifest and Buffering Limits

Manifest Flush Period: The rate (in milliseconds) at which a job's task manifest should be refreshed. Defaults to `100` ms. Minimum `100`, maximum `10000`.

Manifest Max Buffer Size: 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

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`.

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: Defaults to `No`. When toggled to `Yes`, 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

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:

- **Max number of connections**: 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

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.

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.

Proxy Settings

Use proxy env vars: Honors the `HTTP_PROXY/HTTPS_PROXY` environment variables. Defaults to `Yes`.

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

For details about this left tab's **Process count**, **Minimum process count**, and **Memory (MB)** controls, see Sizing and Scaling.

Process count and **Minimum process count** are not configurable on Cribl Edge, where each Edge Node is automatically allocated `1` Worker Process.

The following controls are also available on this tab to optimize Worker Processes' throughput on startup.

Max connections at startup: Maximum number of connections accepted at Worker Process startup. Defaults to `1`. Enter a negative integer for unlimited connections.

Startup throttling duration (ms): Maximum time (in milliseconds) to continue throttling connections after Worker Process startup. Defaults to `10000` ms (10 sec.) Enter `0` to disable throttling.

Load throttle %: Sets a threshold to prevent overwhelming Workers. If 90% of a Worker Process' CPU utilization readings exceed this threshold over one minute, the process will reject new connections until the

CPU load stabilizes. Another process that is below the threshold will accept the connection the next time it is established. Defaults to `0%` (no throttling). Enter a percentage between `1-100` to enable throttling.

You can configure the CPU saturation threshold, but the 90% sampling trigger is not configurable. Also, `_raw stats > cpuPerc` values might diverge from your **Load throttle %** threshold. This is because `cpuPerc` is sampled and averaged once per minute, whereas the **Load throttle %** is evaluated every second, with a rolling 1-minute lookback sample. (These intervals are also not configurable.)

Enable heap snapshots: Defaults to `No`. Toggle to `Yes` 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.

Other Settings

This page's remaining options work essentially the same way as their counterparts in **Settings** > **Global**. Use the following links for details about: logging levels/redactions, access management, security, scripts, and diagnostics.

7.7. Customizing The Interface

7.7.1. 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 1,024 characters. (The displayed banner will truncate characters that cannot fit across one line of a given viewport.)
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 512 characters, but a long label can force-truncate the **Banner message**. A short label can help 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.

7.7.2. Custom Login Page

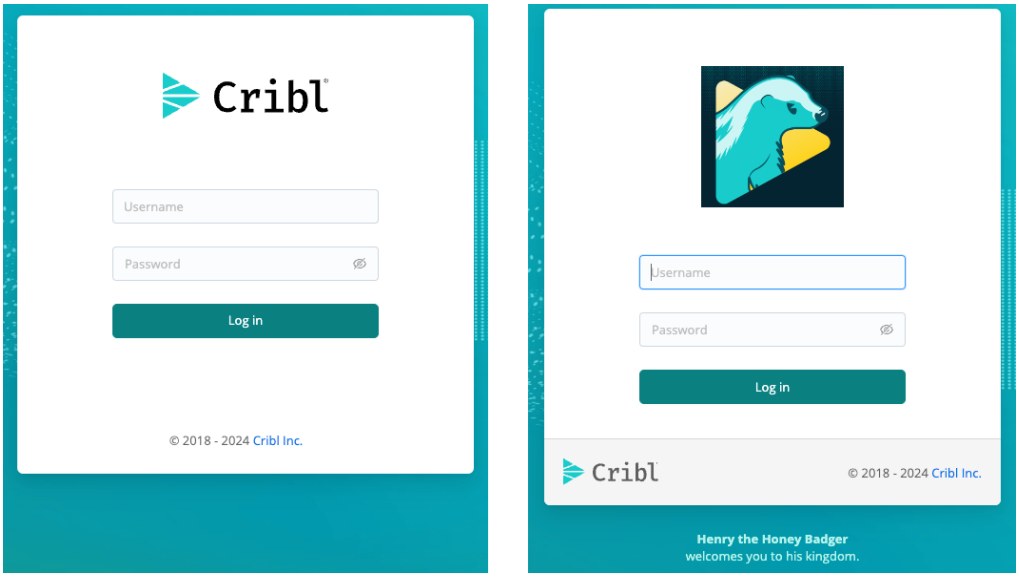
You can customize the look of the login page for all Cribl apps.

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** to `Yes`.
4. In **Logo file**, select an image file that wll serve as a logo. The image must be no larger than 350x350px and under 2MB.
5. In **Content**, edit the provided HTML to modify the text block displayed under the login box.



Built-in and customized login pages

7.8. 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).

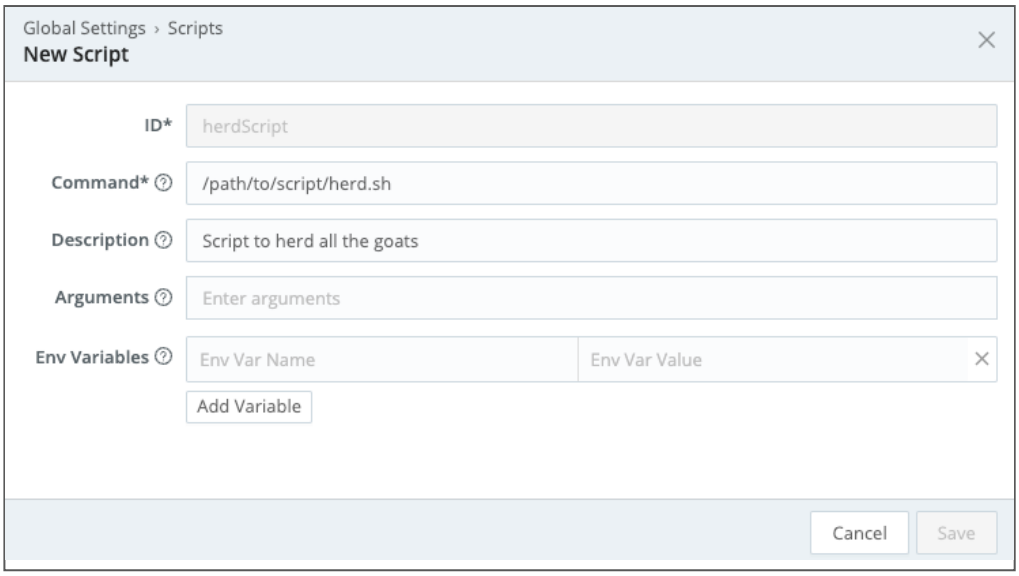
To control script availability, run:

```
# Enable scripts
./cribl limits -s true
# Disable scripts
./cribl limits -s false
```

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.

8. Access Management

Cribl Edge provides a range of access-management features for users with different security requirements. Acccess management serves these common enterprise goals:

- **Security**: Limit the blast radius of inadvertent or intentional errors, by restricting each user's actions to their needed scope within the application.
- **Accountability**: Ensure compliance, by restricting read and write access to sensitive data.
- **Operational efficiency**: Match enterprise workflows, by delegating access over subsets of objects/resources to appropriate users and teams.

Where Can I Find Access Control Details?

See the following topics, according to your needs:

- Authentication: Authenticating users via local basic auth or external options (SSO, Splunk, LDAP).
- Members and Permissions: Fine-grained access control configurable at separate levels (Organization, product, Fleet, and lower-level resources like Cribl Stream Projects and Cribl Search Datasets).
- Teams: Groups of Members who share the same Permissions.
- Local Users: Cribl Edge's original Role-based model for creating users, and for managing their access across a Cribl deployment.
- Roles: Cribl Edge's original RBAC model for managing Roles and Policies, and for assigning them to users.

Which Access Method Should I Use?

Cribl supports both the Members/Permissions and the legacy Users/Roles models, and these models are cross-compatible for many use cases. However, certain purposes require you to choose a specific model:

- **Cribl.Cloud** relies only on Members/Permissions. See Cribl.Cloud Organization-level Permissions starting at Inviting Members, and product- and lower-level Permissions starting at Product-Level Permissions.

  > Cribl.Cloud's Organization-level Permissions include an Owner superuser. This option has no counterpart at the on-prem (customer-managed) Organization level.

- **Stream Projects** and Subscriptions rely only on Members/Permissions. See Project-Level Permissions.
- **GitOps** integration authorization requires the legacy `gitops` Role. This legacy Role has no counterpart Permission.

- **Collectors**: The `collect_all` Role specifically enables creating, configuring, and running Collection jobs on all Stream Worker Groups. This legacy Role has no counterpart Permission.
- **Notifications**: The `notification_admin` Role specifically enables creating and receiving all Notifications. This legacy Role has no counterpart Permission.
- **Search granular resources** (Datasets, Dataset Providers, and search results) can be shared via Members/Permissions. For details, see the Cribl Search Sharing page.

8.1. Authentication

User authentication in Cribl Edge

Cribl Edge supports **local**, **Splunk**, **LDAP**, **SSO/OpenID Connect**, and **SSO/SAML** authentication methods, depending on license type.

Local Authentication

To set up local authentication:

1. In the sidebar, select **Settings**, then **Global**.
2. Under **Access Management**, select **Authentication**.
3. Choose **Local** as authentication **Type**.

You can then manage users through the **Settings** > **Global** > **Access Management** > **Local Users** page.
All changes made to users are persisted in a file located at `$CRIBL_HOME/local/cribl/auth/users.json`.

For Windows Edge Nodes, the changes are persisted in a file located at `C:\ProgramData\Cribl\local\cribl\auth\users.json`.

This is the line format, and note that both usernames and passwords are case-sensitive:

```
{"username":"user","first":"Goat","last":"McGoat","roles":
["admin"],"disabled":"false","passwd":"Yrt0MOD1w8OzyMYB8WMcEleOtYESMwZw2qIZyTvueOE"}
```

The file is monitored for modifications every 60s, and will be reloaded if changes are detected.

Adding users through direct modification of the file is also supported, but not recommended.

If you edit `users.json`, maintain each JSON element as a single line. Otherwise, the file will not reload properly.

Manual Password Replacement

Manual Password Replacement is for on-prem deployments only. For Cribl Edge in Cribl.Cloud, Cribl recommends setting up Fallback Access; if you do need to reset your password, you must contact Cribl.

To manually add, change, or restore a password, replace the affected user's `passwd` key-value pair with a `password` key, in this format: `"password":"<newPlaintext>"`. Cribl Edge will hash all plaintext password(s), identified by the `password` key, during the next file reload, and will rename the plaintext `password` key.

Starting with the same `users.json` line above:

```
{"username":"user","first":"Goat","last":"McGoat","roles":
["admin"],"disabled":"false","passwd":"Yrt0MOD1w8OzyMYB8WMcEleOtYESMwZw2qIZyTvueOE"}
```

...you'd modify the final key-value pair to something like:

```
{"username":"user","first":"Goat","last":"McGoat","roles":
["admin"],"disabled":"false","password":"V3ry53CuR&pW9"}
```

Within at most one minute after you save the file, Cribl Edge will rename the `password` key back to `passwd`, and will hash its value, re-creating something resembling the original example.

Set Worker/Edge Node Passwords

In a Distributed deployment (Edge, Stream), once a Worker/Edge Node has been set to point to the Leader Node, Cribl Edge will set each Worker/Edge Node's admin password with a randomized password that is different from the admin user's password on the Leader Node. This is by design, as a security precaution. But it might lead to situations where administrators cannot log into a Worker/Edge Node directly, and must rely on accessing them via the Leader.

To explicitly apply a known/new password to your Edge Node, you set and push a new password to the Fleet. Here's how, in the Leader Node's UI:

1. On the top bar, select **Products**, and then select **Cribl Edge**.
2. Select the desired Fleet.
3. On the Fleet submenu, select **Fleet Settings**.
4. Select **Local Users**, then expand the desired user.
5. Update the **Password** and **Confirm Password** fields and select **Save**.

Every 10 seconds, the Worker/Edge Node will request an update of configuration from the Leader, and any new password settings will be included.

On Cribl.Cloud, Group Settings for customer-managed hybrid Workers include the above **Local Users** options, but Groups for Cribl-managed Workers in Cribl.Cloud do not. For alternatives, see the Cribl.Cloud Launch Guide and Cribl.Cloud SSO Setup docs.

Authentication Controls

You can customize authentication behavior at **Settings** > **Global** > **General Settings** > **API Server Settings** > **Advanced**. The options here include:

- **Logout on roles change**: If role-based access control is enabled, determines whether users are automatically logged out of Cribl Edge when their assigned Roles change. Defaults to `Yes`.
- **Auth-token TTL**: Sets authentication tokens' valid lifetime, in seconds. Defaults to `3600` (60 minutes = 1 hour); must be at least `1` sec.
- **Session idle time limit**: Sets how long (in seconds) Cribl Edge will observe no user interaction before invalidating user's session tokens. Defaults to `3600` (60 minutes = 1 hour); must be at least `60` sec.
- **Login rate limit**: Sets the number of login attempts allowed over a (selectable) unit of time. Defaults to `2/second`.
- **HTTP headers**: Enables you to specify one or more custom HTTP headers to be sent with every response.

Token Renewal and Session Timeout

Here is how Cribl Edge sets tokens' valid lifetime by applying the **Auth-token TTL** field's value:

- When a user logs in, Cribl Edge returns a token whose expiration time is set to {login time + **Auth-token TTL** value}.
- If the user is idle (no UI activity) for the configured token lifetime, they are logged out.
- As long as the user is interacting with Cribl Edge's UI in their browser, Cribl Edge continually renews the token, resetting the idle-session time limit back by the **Auth-token TTL** value.

The cribl.secret File

When Cribl Edge first starts, it creates a `$CRIBL_HOME/local/cribl/auth/cribl.secret` file. This file contains a key that is used to generate auth tokens for users, encrypt their passwords, and encrypt encryption keys.

Default local credentials are: `admin/admin`

Back up and secure access to this file by applying strict permissions – for example, `600`.

External Authentication

While configuring any external auth method, make sure you don't get locked out of Cribl Edge! Enable the **Fallback on fatal error** or **Allow local auth** toggle until you're certain that external auth is working as intended. If you do get locked out, refer back to Manual Password Replacement for the remedy.

All external auth methods require certain plan/license tiers. For details, see Pricing.

Cribl Edge Roles and role mapping are supported **only** with an Enterprise license. With a Standard license, all your external users will be imported to Cribl Edge in the `admin` Role.

This topic covers the following external authentication providers:

- Splunk Authentication
- LDAP Authentication

Authentication for Single Sign-On

Each authentication method that Cribl Edge supports for on-prem SSO has its own separate topic:

- SSO/OpenID Connect Authentication
- SSO/SAML Authentication

For deployments in Cribl.Cloud, see Cribl.Cloud SSO Setup.

Splunk Authentication

Splunk authentication is very helpful when deploying Cribl Edge in the same environment as Splunk. This option requires the user to have Splunk `admin` role permissions. To set up Splunk authentication:

Navigate to **Settings** > **Global** > **Access Management** > **Authentication** > **Type** and select **Splunk**. This exposes the following controls.

- **Host**: Splunk hostname (typically a search head).
- **Port**: Splunk management port (defaults to `8089`).
- **SSL**: Whether SSL is enabled on Splunk instance that provides authentication. Defaults to `Yes`.
- **Validate certificates**: When enabled, Cribl Edge will reject any certificate that the Certificate Authority (CA) for your system does not authorize. Although this control is off by default, Cribl strongly recommends that you enable it, except when you must allow untrusted (for example, self-signed) certificates.

- **Fallback on fatal error**: Attempt local authentication if Splunk authentication is unsuccessful. Defaults to `No`. If toggled to `Yes`, Cribl Edge will attempt local auth only **after** a failed Splunk auth. Selecting `Yes` also exposes this additional option:
    - **Fallback on bad login**: Attempt local authentication if the supplied user/password fails to log in on Splunk. This similarly defaults to `No`.

The Splunk search head does not need to be locally installed on the Cribl Edge instance.

For distributed deployments with an Enterprise License, the next (and final) step is to configure Role Mapping settings.

LDAP Authentication

You can set up LDAP authentication as follows:

Navigate to **Settings** > **Global** > **Access Management** > **Authentication** > **Type**, and select **LDAP**. This exposes the following controls.

- **Secure**: Enable to use a secure LDAP connections (`ldaps://`). Disable for an insecure (`ldap://`) connection.
- **LDAP servers**: List of LDAP servers. Each entry should contain `host:port` – for example, `localhost:389`.
- **Bind DN**: Distinguished Name of entity to authenticate with LDAP server – for example, `'cn=admin,dc=example,dc=org'`.
- **Password**: Distinguished Name password used to authenticate with LDAP server.
- **User search base**: Starting point to search LDAP for users – for example, `'dc=example,dc=org'`.
- **Username field**: LDAP user search field, such as `cn` or `(cn (or uid)`. For Microsoft Active Directory, use `sAMAccountName` here.
- **User search filter**: LDAP search filter to apply when authenticating users. Optional, but recommended: If empty, all users in your domain will be able to log in with the default Cribl Role. This example would enable only users in the `StreamAdmins` group to log in – they would then be mapped to any Roles specified in Role Mapping Settings:
  `(&(objectClass=user)(memberof=CN=StreamAdmins,OU=Groups,DC=cribl,DC=io))`
  This simpler filter example would enable anyone in a group called admin," who does **not** have a department attribute that starts with "123," to log in:
  `(&(group=admin)(!(department=123*)))`
- **Group search base**,
  Group member field,
  Group membership attribute,

Group search filter,
Group name field: These settings are used only for LDAP authentication with role-based access control. See Role-Based LDAP Authentication, below.

- **Connection timeout (ms)**: Defaults to `5000`.
- **Reject unauthorized**: Valid for secure LDAP connections. Set to `Yes` to reject unauthorized server certificates.
- **Fallback on fatal error**: Attempt local authentication if LDAP authentication is down or misconfigured. Defaults to `No`. If toggled to `Yes`, local auth will be attempted only **after** a failed LDAP auth. Selecting `Yes` also exposes this additional option:
    - **Fallback on bad login**: Attempt local authentication if the supplied user/password fails to log in on the LDAP provider. Defaults to `No`.

For distributed deployments with an Enterprise License, the next (and final) step is to configure Role Mapping settings.

Role-Based LDAP Authentication

When configuring LDAP authentication with role-based access control (RBAC), you **must** use the following settings to import user groups. (The UI does not enforce filling these fields. When using LDAP without roles, ignore them.)

- **Group search base**: Starting point to search LDAP for groups – for example, `dc=example,dc=org`.
- **Group member field**: LDAP group search field – for example, `member`.
- **Group membership attribute**: Attribute name of LDAP user object. Determines group member attribute's value, which defines group's allowed users. This field accepts `dn`, `cn`, `uid`, or `uidNumber`. If none of these are specified, falls back to `dn`.
- **Group search filter**: LDAP search filter to apply when retrieving groups for authorization and mapping to Roles – for example, `(&(cn=cribl*)(objectclass=group))`.
- **Group name field**: Attribute used in objects' DNs that represents the group name – for example, `cn`. Cribl Edge does not directly read this attribute from group objects; rather, it must be present in your groups' DN values. Match the attribute name's original case (upper, lower, or mixed) when you specify it in this field. In particular, Microsoft Active Directory requires all-uppercase group names – for example, `CN`.

For distributed deployments with an Enterprise License, the next (and final) step is to configure Role Mapping settings.

Role Mapping Settings

This section is displayed only on Distributed deployments (Edge, Stream) with an Enterprise License. For details on mapping your external identity provider's configured groups to corresponding Cribl Edge user access Roles, see External Groups and Roles. The controls here are:

- **Default Role**: Default Cribl Edge Role to assign to all groups not explicitly mapped to a Role.
- **Mapping**: On each mapping row, enter an external group name (case-sensitive) on the left, and select the corresponding Cribl Edge Role in the right drop-down list. Click **Add Mapping** to add more rows.

8.2. Members And Permissions

Define and manage fine-grained access control across Cribl products and resources

---

Members and Permissions enable fine-grained access control and authorization. You can assign different access and capabilities (Permissions) to each user (Member) at multiple levels:

- Organization.
- Product: Stream, Edge, Search, or Lake.
- Group/Fleet.
- Resource: Cribl Stream Project, Cribl Search Dataset or Dataset Provider, and so on.

Availability

Permission-based access control is available on:

- On-prem Distributed deployments (Stream, Edge) with certain plan/license tiers.
- All Cribl.Cloud deployments, at the Organization level. (Cribl.Cloud with certain plan/license tiers also offers granular access control on products, Fleets, and lower-level resources.)

On on-prem Single-instance deployments, or Distributed deployments with other licenses, all users will have full `Admin`-level privileges.

Initial Permissions

When you first deploy Cribl Edge with the above prerequisites, you will be granted the Organization-level `Admin` Permission. Using this Permission, you can then assign additional Permissions to yourself and other Members, as outlined below in Organizations and Assigning Group-/Fleet-Level Permissions.

Cross-Compatibility

Members and Permissions are available as the successors to Cribl's original role-based access control (RBAC) model of Local Users and Roles/Policies. The earlier model is still supported across most of the Cribl product suite.

However, Cribl.Cloud invitations and Stream Projects have fully transitioned to the new model. For detailed differences between the two access control systems, see Which Access Method Should I Use?.

Known Issue

Existing Local Users display in **Settings** > **Global** > **Members and Teams** with the `No Access` Permission even if they've been assigned a higher Role. This is a display-only bug: These users' original Roles still function as configured. For details and fix progress, please see Known Issues.

Organizations

The Members and Permissions model uses the concept of Organization as a container for the deployment of a whole suite of Cribl products (Stream, Edge, and Cloud-only Search and Lake). To access the Organization-level Members UI as an Admin:

1. In the sidebar, select **Settings**, then **Global**.
2. Under **Access Management**, select **Members and Teams**.
3. Here, you can select an existing Member's row to change their access Permissions or other details, or select **Add Member** to grant a new user access to your Cribl Organization.

Invite Members

To invite new Members to your Organization:

1. In the sidebar, select **Settings**, and then select **Global**.
2. Under **Access Management** submenu, select **Members and Teams**.
3. Select **Add Member**.
4. Enter the new Member's information, email, and password.
5. Under **Workspace permission**, select the permission you want to grant to the new Member for the whole Organization.
6. If you chose the **User** permission, configure access for each Cribl product separately. An **Admin** permission automatically grants admin access to all products.
7. When finished, select **Save**.

For Cribl.Cloud's counterpart to this on-prem UI for managing Organization-level Permissions, see Inviting Members.

Members and Local Users

Members and Local Users are interchangeable: Members can be reconfigured using legacy Roles instead of Permissions, and anyone you add to **Local Users** will also show in **Members**.

Organization- and Product-Level Permissions

Each newly created Member starts out with **User** Permission at the Organization level, and with the self-explanatory **No Access** Permission on each product. You can use the toggles to assign the following Permission levels.

For Cribl.Cloud Organization-level Permissions, see Cribl.Cloud docs' Member Permissions section.

Organization-Level Permissions

Permission	Description
User	The most basic Permission. At the Organization level, gets the Member into the system, without conferring any access to peer Members, products, or lower-level resources. (Admins can freely assign these Members varying Permissions at lower levels.)
Admin	This is a superuser Permission. At the Organization level, allows creating, viewing, updating, and deleting all Members. (Automatically inherits Product-level `Admin` Permissions and resource-level `Maintainer` Permissions, which confer comparably broad capabilities.)

For Stream and Edge, the Product-level Permissions below apply to both Cribl.Cloud and on-prem deployments. For Cribl Search's more-specific Product-level Permissions, see Search Member Permissions. For Cribl Lake's Product-level Permissions, see Lake Permissions.

Product-Level Permissions (Stream and Edge)

Permission	Description
User	The most basic Permission. At the Product level, makes the Member assignable to Fleets and resources, with no initial access to any.
Read Only	At the Product level, this Permission is designed specifically to enable support personnel to help other users by viewing (but not modifying) their configurations. Allows viewing all Members, Fleets, Settings, Leader commits, and legacy Local Users and Roles, with no configuration capabilities. (A `Read Only` Permission at the Product level automatically inherits `Read Only` Permissions on all Fleets and lower-level resources.)

Permission	Description
Editor	Allows viewing all Groups and Monitoring pages. (Configuring these options requires `Admin`-level permission on the product and/or Fleet.) Automatically inherits the `Editor` Permission on Fleets, and the `Maintainer` Permission on lower-level resources.
Admin	This is a superuser Permission at the Product level. It allows creating, viewing, updating, and deleting all Fleets and resources; managing Fleet Mappings; adding, updating, and restarting Edge Nodes; and managing Notifications and Notification targets. (Automatically inherited from the Organization-level `Admin` Permission in on-prem deployments, and from the Org-level `Owner` Permission in Cribl.Cloud deployments. Automatically inherits `Admin` Permissions on all Fleets and `Maintainer` Permissions on lower-level resources.)

Once Members are in your Organization, you can assign or reassign the above Organization- and Product-level Permissions. Return to **Settings** > **Global** > **Members and Teams** and select the Member's row to reopen the same configuration modal.

Group- and Fleet-Level Permissions

Cribl Edge Permissions at the Fleet level generally mirror those at the Product level:

Permission	Description
User	The most basic Permission. Gets the Member into the Fleet, where Admins can then assign them access on individual resources. (Unless Members have a higher Permission at the Product level, they have no initial access to Fleets or resources.)
Read Only	Designed for support personnel. Allows viewing all Fleet-level Settings, encryption keys, certificates, secrets, scripts, Sources, Destinations, Pipelines, Packs, Routes, QuickConnect connections, Knowledge objects, Notifications and Notification targets, Edge Subfleets and their Settings, and Stream Projects and Subscriptions, with no configuration capabilities. (Automatically inherited from a Product-level `Read Only` Permission. Automatically inherits `Read Only` Permissions on lower-level resources.)
Editor	Allows creating, viewing, updating, and deleting all Fleet-level encryption keys, certificates, secrets, scripts, Sources, Destinations, Pipelines, Packs, Routes, QuickConnect connections, Knowledge objects, and Notifications and Notification targets. Allows committing configuration changes, but not deploying them. (An `Editor` Permission at the Product level automatically inherits `Editor` Permissions on all Fleets.)
Admin	Superuser Permission. Allows creating, viewing, updating, and deleting all Fleet-level access management (Members' Permissions), Settings, encryption keys, key management system (KMS) settings, certificates, secrets, scripts, Sources, Destinations, Pipelines, Packs, Routes, QuickConnect connections, Knowledge objects, Notifications and Notification targets, and Stream Projects/Subscriptions.

Permission	Description
	Allows adding, updating, and restarting Edge Nodes. Allows committing and deploying configuration changes. On Edge, provides CRUD capabilities on Subfleets' Settings and access management identical to those on the parent Fleet. (Automatically inherited from the Product-level `Admin` Permission. Automatically inherits `Maintainer` Permissions on lower-level resources.)

Assigning Group-/Fleet-Level Permissions

To assign Permissions on a Fleet:

1. In the sidebar, select **Settings**.
2. Select **Global**, then **Members and Teams**.
3. Make sure this Member has a **Workspace permission** level sufficient to support the Permission you want to assign on the Fleet.
4. On the **Settings** submenu, select **Stream/Edge**, then **Members and Teams**.
5. Select each applicable Member's row to open their **Member Details** drawer.
6. Grant the Member the Permission level you want on this Fleet.

Assigning Group-/Fleet-Level Permissions in only available in on-prem deployments, not in Cribl.Cloud.

Resource-Level Permissions

Certain Cribl products provide Permission-based access to particular resources.

Stream Projects

Stream Projects and Subscriptions rely entirely on the Members/Permissions access control system. These enable you to assign different users different levels of access on individual Projects.

The 4.1.x `project_user` Role and `ProjectSourceSubscribe` Policy, which provided access to all Projects, are retired in Cribl Edge 4.2.x and later. **This is a breaking change.** If you are upgrading with Projects configured in v.4.1.x, those Projects' users will be visible in Stream as Members, and you'll need to assign them new Permissions.

For details, see Adding Users to Projects.

Search Resources

For Permissions on Search resources, see the Search docs' Sharing topic.

Other Resources

Access to certain resources can be managed only via legacy Local Users and Roles.

GitOps integration: Authorization to enable and manage GitOps requires the legacy `gitops` Role. This legacy Role has no counterpart Permission.

Collectors: The `collect_all` legacy Role specifically enables creating, configuring, and running Collection jobs on all Stream Worker Groups. This Role has no counterpart Permission.

Notifications: The `notification_admin` legacy Role specifically enables creating and receiving all Notifications. This legacy Role has no counterpart Permission.

Inheritance

Members' Permissions at certain levels determine the Permissions that Admins can assign them at lower levels. Here is the current inheritance scheme:

- Members with the `Admin` Permission at the Organization level will be locked to the `Admin` Permission on Products, to the `Editor` Permission on Fleets, and to the `Maintainer` Permission on lower-level resources.
- Members with the Product-level `Editor` Permission will be locked to the `Editor` Permission on Fleets, and to the `Maintainer` Permission on lower-level resources.
- Members with the `User` Permission at each level can be assigned varying Permissions at the next level down. **`User` is the most malleable Permission.**
- On Stream Projects, the `Maintainer` Permission is available **only** to Members with higher-level `Admin` or `Editor` Permissions. (They're automatically assigned this Permission via inheritance.)
- Search resources are more flexible: Here, the `Maintainer` Permission can be shared with Members who have the `User` Permission at the Product level.
- Members with the Product-level `Read Only` Permission will be locked to the `Read Only` Permission on Fleets and on lower-level resources.

8.3. Teams

Teams are groups of Members who share the same Organization, Workspace, and product-level permissions. This allows you to efficiently manage access control by assigning permissions to the Team rather than configuring each Member individually.

Teams are available only in distributed deployments or in Cribl.Cloud.

Create a Team

To create and configure a Team:

Cribl.Cloud | On-Prem Deployment

1. On the top bar, select **Products**, and then select **Cribl**.
2. In the sidebar, select **Organization**, then **Members & Teams**.
3. Go to the **Teams** tab and select **Create Team**.
4. Provide a meaningful **Name** and optionally a **Description**.
5. If you have SSO configured in your Organization, in **Mapping ID**, you can provide an external ID for the Team. Use it when integrating with SSO to map Workspace roles with IDP groups.
6. Select the desired Workspace and access to assign to the Team. **Member** allows granular product-level permissions.
7. In **Team Members**, select the Members you want to add, then select **Add**. A Member must already exist to be added to a Team.
8. The table is updated to show the Members you've just added.
9. When finished, select **Save**.

The main Teams page displays the Teams configured on your Organization. You can edit them by clicking their row, or delete them as needed from the **Actions** column.

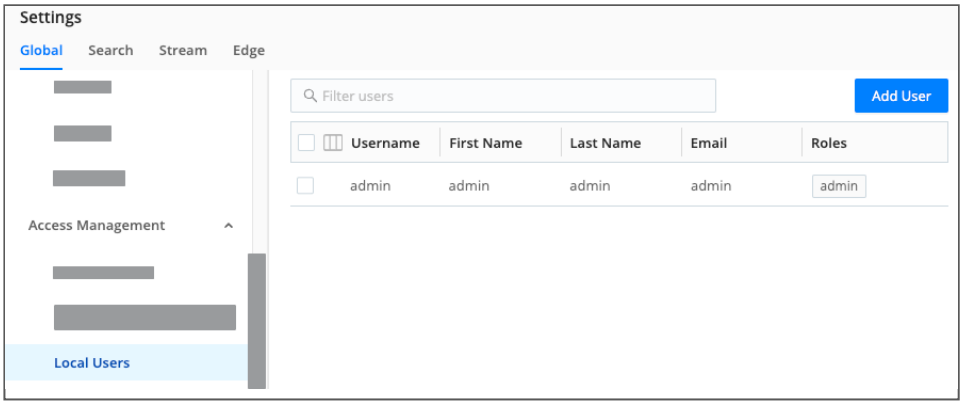
8.4. Local Users

This page covers how to create and manage Cribl Edge users, including their credentials and (where enabled) their access roles. These options apply if you're using the **Local** Authentication type, which is detailed here.

Creating and Managing Local Users

To create a new Local User:

1. In the sidebar, select **Settings**, then **Global**.
2. Under **Access Management**, select **Local Users**. The **Local Users** page will initially show only the default `admin` user. You are operating as this user.



Managing users

3. Select **Add User**.
4. Provide the new User with basic information and set their **Password**. Usernames and passwords are case-sensitive.
5. To prevent the user from currently logging in, toggle **Enabled** to **No**.

Logged-in users can change their own Cribl Edge password from the **Local Users** page, by clicking on their own row to open a **Local Users** modal that manages their identity.

Password Rules

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 a new password, the new password 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.

Adding Roles

If you've enabled role-based access control you can use the modal's bottom **Roles** section to assign access Roles to this new or existing user.

For details, see Roles. Role-based access control can be enabled only on distributed deployments (Edge, Stream) with certain plan/license tiers. With other license types and/or single-instance deployments (Edge, Stream), all users will have full administrative privileges.

Select **Add Role** to assign each desired role to this user. The options on the **Roles** drop-down reflect the Roles you've configured at **Settings** > **Global** > **Access Management** > **Roles**.

Note that when you assign multiple Roles to a user, the Roles' permissions are additive: This user is granted a superset of the highest permissions contained in all the assigned Roles.

By default, Cribl Edge will log out a user upon a change in their assigned Roles. You can defeat this behavior at **Settings** > **Global** > **General Settings** > **API Server Settings** > **Advanced** > **Logout on roles change.**

8.5. Roles

Define and manage access-control Roles and Policies

The Roles/Policies model exists in parallel with a more flexible Members/Permissions model, which will eventually replace it. Cross-compatible Default Roles and Default Policies support customers who still choose to configure Local Users with Roles and Policies. Your configured Local Users appear interchangeably in the new Members UI, and vice versa.

See Which Access Method Should I Use? for detailed differences between the two access control systems.

Availability

Role-based access control is available only on distributed deployments (Stream, Edge) with a certain plan/license tier.

On single-instance deployments or distributed deployments with other licenses all users will have full administrative privileges.

Known Issue

Existing Local Users display in **Settings** > **Global** > **Members and Teams** with the `No Access` Permission even if they've been assigned a higher Role. This is a display-only bug: These users' original Roles still function as configured. For details and fix progress, please see Known Issues.

RBAC Concepts

Cribl Edge's RBAC mechanism is designed around the following concepts, which you manage in the UI:

- **Roles**: Logical entities that are associated with one or multiple **Policies** (groups of permissions). You use each Role to consistently apply these permissions to multiple Cribl Edge **users**.
- **Policies**: A set of **permissions**. A Role that is granted a given Policy can access, or perform an action on, a specified Cribl Edge object or objects.
- **Permissions**: Access rights to navigate to, view, change, or delete specified **objects** in Cribl Edge.
- **Users**: You map Roles to Cribl Edge users in the same way that you map **user groups** to users in LDAP and other common access-control frameworks.

Users are independent Cribl Edge objects that you can configure even without RBAC enabled. For details, see Local Users.

Cribl Edge Roles can be integrated with external authorization/IAM mechanisms, such as LDAP and OIDC and mapped to their respective groups, tags, etc.

Using Roles

Cribl Edge ships with a set of default Roles, which you can supplement.

Default Roles

These Roles ship with Cribl Edge by default.

Organization-Level Roles

Note that some of the Roles below have no counterpart Permission in Cribl's newer Members/Permissions model.

Name	Description	Permission Equivalent
admin	Superuser – authorized to do anything in the system.	Organization Admin
gitops	Ability to sync the Cribl Edge deployment to a remote Git repository.	N/A
notification_admin	Read/write access to all Notifications.	N/A
user	Default Role that gets only a home/landing page to authenticate. This is a fallback for users who have not yet been assigned a higher Role by an admin.	Organization User
project_user	Read/write access to the simplified Stream Projects UI and related data Subscriptions. **Deprecated as of v.4.2.** Instead assign **Editor**, as the most comparable new Project-level Permission. The more-permissive **Maintainer**, and the more-restrictive **Read Only**, are also available Permissions.	Project Editor

Stream Roles

Name	Description	Permission Equivalent
stream_user	Basic Role for Stream.	Stream User.
stream_reader	Allows viewing all Members, Worker Groups, Settings, Leader commits, and legacy Local Users and Roles, with no configuration capabilities.	Stream Read Only.
stream_editor	Allows viewing all Groups and Monitoring pages.	Stream Editor.
stream_admin	Superuser Role at the Product level	Stream Admin.

Edge Roles

Name	Description	Permission Equivalent
edge_user	Basic Role for Edge.	Edge User.
edge_reader	Allows viewing all Members, Fleets, Settings, Leader commits, and legacy Local Users and Roles, with no configuration capabilities.	Edge Read Only.
edge_editor	Allows viewing all Groups and Monitoring pages.	Edge Editor.
edge_admin	Superuser Role at the Product level.	Edge Admin.

Fleet–Level Roles

Name	Description	Permission Equivalent
owner_all	Read/write access to (and Deploy permission on) all Fleets.	N/A
editor_all	Read/write access to all Fleets.	N/A
reader_all	Read-only access to all Fleets.	N/A

Name	Description	Permission Equivalent
collect_all	Ability to create, configure, and run Collection jobs on all Worker Groups.	N/A

Cribl Search Roles

Name	Description	Permission Equivalent
search_user	Basic Role for Cribl Search.	Cribl Search User.
search_editor	Manage Datasets, Dataset Providers, Dashboards and settings.	Cribl Search Editor.
search_admin	Superuser Role at the Product level.	Cribl Search Admin.

Cribl **strongly recommends** that you do not edit or delete these default Roles. However, you can readily clone them, and modify the duplicates to meet your needs.

Initial Installation or Upgrade

When you first install Cribl Edge with the prerequisites to enable RBAC (Distributed deployment with a certain plan/license tier), you will be granted the **admin** Role. Using this Role, you can then define and apply additional Roles for other users.

You will similarly be granted the **admin** Role upon upgrading an existing Cribl Edge installation from pre-2.4 versions to v. 2.4 or higher. This maintains backwards-compatible access to everything your organization has configured under the previous Cribl Edge version's single Role.

Adding and Modifying Roles

To add a new Role:

1. In the sidebar, select **Settings**, then **Global**.
2. Under **Access Management**, select **Roles**.
3. Select **Add Role**.
4. Provide the Role **Name**.
5. Select **Policies** for this Role.

Adding and Removing Policies

The **Policies** section is an expandable table. In each row, you select a Policy using the left drop-down, and apply that Policy to objects (i.e., assign permissions on those objects) using the right drop-down.

Let's highlight an example from the above screen capture of Cribl Edges built-in Roles: The `editor_all` Role has the `GroupEdit` Policy, with permission to exercise it on any and all Fleets (as indicated by the `*` wildcard).

Policies on the left, objects on the right

To add a new Policy to a Role:

1. Select **Add Policy** to add a new row to the **Policies** table.
2. Select a Policy from the left column drop-down.
3. Accept the default object on the right; or select one from the drop-down.

To modify an already-assigned Policy, just edit its row's drop-downs in the **Policies** table.

To remove a Policy from the Role, select the **X** icon at right.

In all cases, click **Save** to confirm your changes and close the modal.

Default Policies

In the **Policies** table's left column, the drop-down offers the following default Policies:

Fleet-Level Policies

Name	Description	Permission Equivalent
`GroupRead`	The most basic Fleet-level permission. Enables users to view a Fleet and its configuration, but not modify or delete the config.	Fleet-level Read Only.
`GroupEdit`	Building on `GroupRead`, grants the ability to also change and commit a Fleet's configuration.	Fleet-level Editor.
`GroupFull`	Building on `GroupEdit`, grants the ability to also deploy a Fleet.	Fleet-level Admin.

Name	Description	Permission Equivalent
`GroupCollect`	Grants the ability to create, configure, and run Collectors on a Worker Group.	N/A
`GroupUser`	Access Fleet.	Fleet-level User.

Project-Level Policies

Name	Description	Permission Equivalent
`ProjectMaintain`	Grants ability to edit or delete the Project and its settings.	Project-level Maintainer.
`ProjectRead`	Can configure connections among the Project's Subscriptions, Packs, and Destinations, but cannot modify or delete these resources.	Project-level Read Only.
`ProjectEdit`	Can view Project and Subscription settings and connections, but not modify or delete them.	Project-level Editor.

Product-Level Policies

Name	Description	Permission Equivalent
`ProductUser`	Makes the Member assignable to Worker Groups and resources, with no initial access to any.	Product-level User.
`LimitedProductUser`	Similar to `ProductUser`, but omits the ability to read or act on all the endpoints within a Fleet.	N/A
`ProductAdmin`	Superuser Permission at the Product level.	Product-level Admin.

Search Policies

Name	Description	Permission Equivalent
`DatasetMaintain`	Full access to Dataset configuration.	Search Datasets @Maintainer.
`DatasetRead`	Can view and search, but not modify Dataset configurations.	Search Datasets @Read Only.
`DatasetProviderMaintain`	Full access to Dataset Provider configuration.	Search Dataset Providers @Maintainer.
`DatasetProviderRead`	Can view and search, but not modify Dataset Provider configurations.	Search Dataset Providers @Read Only.
`SearchUser`	Search data and view results shared with the user	Search Product-level @User.
`SearchMaintainer`	Manage Datasets, Dataset Providers, Dashboards and settings.	Search Product-level @Editor.
`DashboardMaintain`	Can edit and delete a Search dashboard.	Search dashboard @Maintainer.
`DashboardRead`	Can view and use a Search dashboard.	Search dashboard @Read Only.

General Policies

Name	Description	Permission Equivalent
`*` (wildcard)	Grants **all** permissions on associated objects.	N/A

Internal Policies

The following policies are internal building blocks for Product-specific Policies. Do not add them directly to Roles.

Name	Description	Permission Equivalent
`Product`	N/A	N/A
`BaseProductUser`	N/A	N/A

Name	Description	Permission Equivalent
`MaintainBase`	N/A	N/A
`SearchBase`	Similar to `SearchUser`: Can view Search resources shared with the Member.	N/A

Use Policies As-Is

By design, the default Policies that ship with Cribl Edge cannot be modified via the UI or API. Do not attempt to modify them by other means. Breaking the built-in model could undermine your intended access-control protections, opening your deployment and data to security vulnerabilities.

Objects and Permissions

In the **Policies** table's right column, use the drop-down to select the Cribl Edge objects on which the left column's Policy will apply. The objects available for selection are specific Fleets, or a wildcard representing all Fleets. For example:

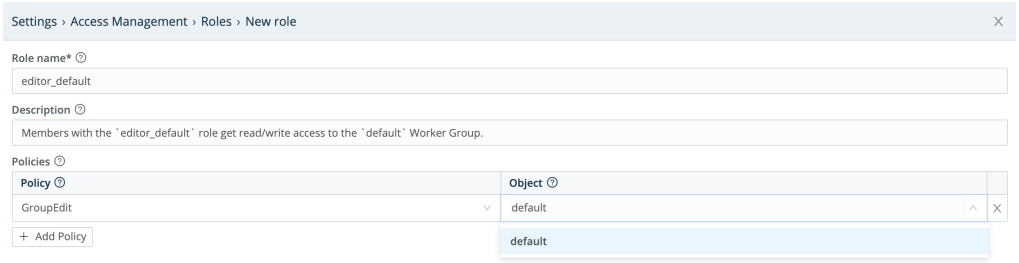
- `Worker Group <id>`
- `NewGroup2`
- `default` (Fleet)
- `*` (all Fleets)

Extending Default Roles

Here's a basic example that ties together the above concepts and facilities. It demonstrates how to add a Role whose permissions are restricted to a particular Fleet.

Here, we've cloned the `editor_all` Role that we unpacked above. We've named the clone `editor_default`.

We've kept the `GroupEdit` Policy from `editor_all`. But in the right column, we're restricting its object permissions to the `default` Fleet that ships with Cribl Edge.



Cloning a default Role

You can readily adapt this example to create a Role that has permissions on an arbitrarily named Fleet of your own.

Roles and Users

Once you've defined a Role, you can associate it with Cribl Edge users. For details, see Local Users.

Note that when you assign multiple Roles to a given user, the Roles' permissions are additive: This user is granted a superset of all the permissions contained in all the assigned Roles.

By default, Cribl Edge will log out a user upon a change in their assigned Roles. You can defeat this behavior at **Settings** > **Global** > **General Settings** > **API Server Settings** > **Advanced** > **Logout on roles change**.

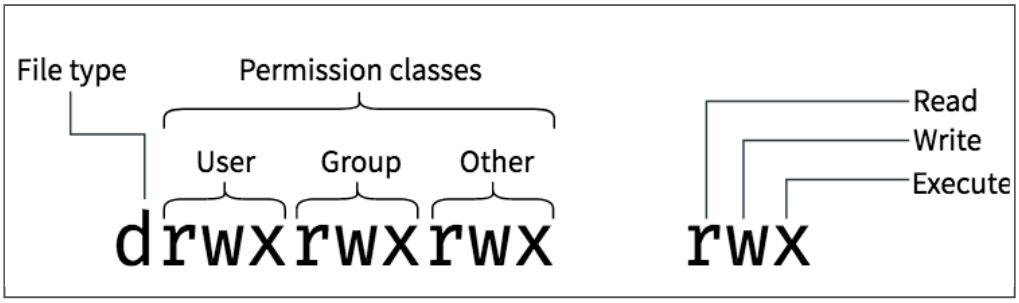
External Groups and Cribl Edge Roles

You can map user groups from external identity providers (LDAP, Splunk, or OIDC) to Cribl Edge Roles, as follows:

1. In the sidebar, select **Settings**, then **Global**.
2. Under **Access Management**, select **Authentication**
3. Choose **LDAP**, **Splunk**, or **OpenID Connect** as authentication **Type**.
4. Configure your identity provider's connection and other basics. (For configuration details, see the appropriate Authentication section.)
5. Under **Role Mapping**, first select a Cribl Edge **Default role** to apply to external user groups that have no explicit Cribl Edge mapping defined below.
6. Next, map external groups as you've configured them in your external identity provider (left field below) to Cribl Edge Roles (right drop-down list below).
7. To map more user groups, select **Add Mapping**.
8. When your configuration is complete, select **Save**.

8.6. Using 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
```

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
```

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>
```

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::---
```

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
```

For Red Hat-based tools using yum:

```
yum install acl
```

8.7. SSO

8.7.1. Cribl.Cloud SSO

8.7.1.1. Cribl.Cloud SSO Setup

With a Cribl.Cloud account on certain plan/license tiers, you can set up a Single Sign-On (SSO) integration between your identity provider and your Cribl.Cloud portal.

The general steps to set up a Single Sign-On (SSO) integration between your identity provider and your Cribl.Cloud portal are:

1. In your identity provider (IDP), configure user groups that map to Cribl.Cloud's Teams or to predefined Roles.
2. In your IDP, create an OIDC or SAML application.

   If creating an OIDC application, you must use backchannel authentication. Cribl.Cloud does not support front-channel authentication via OIDC.
3. Submit your app's configuration details to Cribl on your Cribl.Cloud portal's **Organization** > **SSO Management** page. (This will complete your SSO setup on the Cribl side.)
4. In your IDP, assign groups to your users, matching the Role that each group of users should have in Cribl.Cloud.
5. In your IDP, assign the OIDC/SAML app to the Organization's owner, and to other Cribl.Cloud users.

For detailed instruction on how to configure SSO using different IDPs, see:

- SSO with Okta and OIDC
- SSO with Okta and SAML
- SSO with Microsoft Entra ID and SAML

Additionally, refer to SSO Troubleshooting for help in resolving issues you may encounter when setting up SSO in Cribl.Cloud.

8.7.1.2. Configuring SSO Groups

In your IDP (identity provider), configure user groups that match Permissions for both Cribl.Cloud Organizations and individual products.

Linking IDP Groups with Teams

You can use Teams to automatically connect groups of Members with IDP users. This allows you to grant IDP users Workspace- or Product-level permissions.

For each Team, you can specify a **Mapping ID** to use as an external ID. In your IDP, create a group with the same ID to connect it to the Team and in this way automatically grant IDP users access with Workspace-level permissions.

To create a Team with an external ID, refer to Create a Team. To add an external ID to an existing Team:

1. On the top bar, select **Products**, and then select **Cribl**.
2. In the sidebar, select **Organization**, then **Members & Teams**, and then **Teams**.
3. Select the Team you want to configure.
4. In **Mapping ID**, provide an external ID for the team.
5. Confirm with **Save**.

IDP Group Naming

If you are not using Teams, or want to grant Organization-level permissions to IDP users, you can define groups that will be mapped to Cribl Roles based on their names.

In such a case, the names you define for groups in your IDP must include `Cribl` and the Role name (`Owner`, `Admin`, or `User`). They can include `Organization` or product name (`Stream`, `Edge`, or `Search`). If they don't contain a product name, the group is treated as an Organization name.

Even though IDP group names without Organization will be treated as Organization-level permissions, we recommend keeping Organization in the name for clarity.

You can use either the open or the closed format (with or without spaces) in group names. You can freely add prefixes or suffixes to Group Names that follow the formats in the examples above. Cribl will ignore these additions when mapping IDP groups to Cribl Roles. Examples:

- `SOME-LABEL-12345-CriblOrganizationOwner`

- `CriblOrganizationEditor-420`

Cribl.Cloud Role/Permission	IDP Group Name
Owner	Cribl Organization Owner /or/ CriblOrganizationOwner
Admin	Cribl Organization Admin /or/ CriblOrganizationAdmin
User	Cribl Organization User /or/ CriblOrganizationUser
Editor	**(Deprecated, will be mapped to Admin)** Cribl Organization Editor /or/ CriblOrganizationEditor
Read Only	**(Deprecated, will be mapped to User)** Cribl Organization Read Only /or/ CriblOrganizationReadOnly

Default Product Permissions

When you map external users to your Cribl Organization, their initial product-level Permissions follow a different inheritance pattern than Members configured within Cribl. This is to avoid downgrading product-level Permissions that Organization-level `User`s might already have.

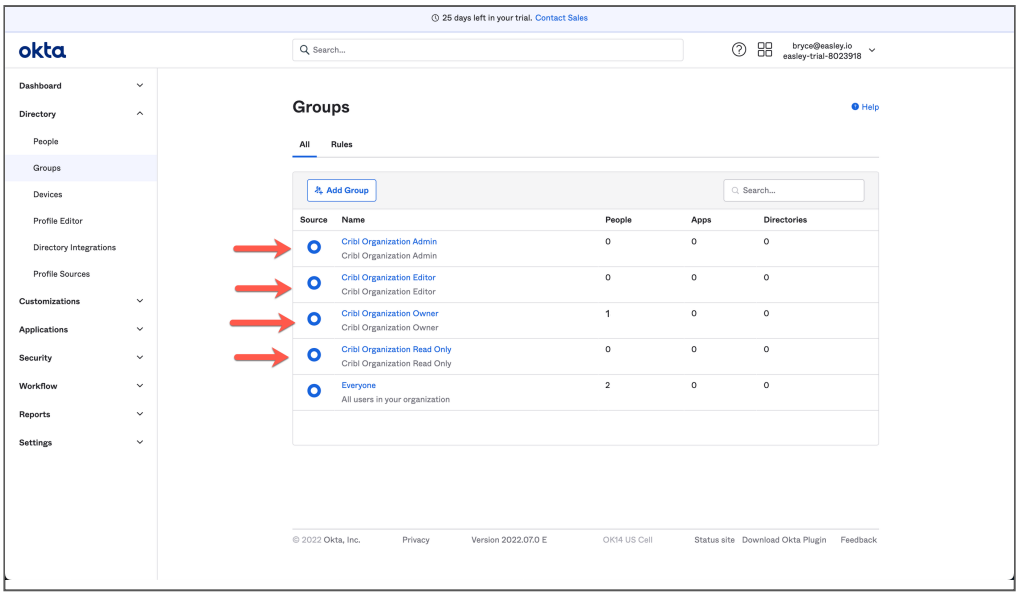
The defaults for mapped users are:

- Organization `Owner` or `Admin` inherits `Admin` Permission on all products.
- Organization `User` inherits `User` Permission on all products (except Cribl Lake, which inherits `No Access`).
- Organization `Editor` (deprecated) inherited `editor` legacy Roles on all products.
- Organization `Read Only` (deprecated) inherited `Read Only` Permission on Stream and Edge, and `User` Permission on Search.

Group Configuration Better Practices

- A Cribl.Cloud Organization's Owner Role can be shared and transferred among multiple users. This facilitates gradual ownership transfers, corporate reorganizations, and other scenarios.
- Those users should be in both the Owner and Admin groups in your IDP. (This enables them to acquire all needed permissions across Cribl's two corresponding Roles.)
- Aside from dual-assigning the Owners, you should assign every other user only one group in your IDP. (Cribl's Admin and Editor Roles include all the permissions of the Roles below them.)

Here’s an example of how groups configuration (at an early stage) might look in Okta’s UI:



Mapping Okta groups to Cribl.Cloud Roles

8.7.1.3. SSO With Okta And OIDC (Cribl.Cloud)

This page presents a walkthrough of setting up an OIDC SSO, using Okta as the example.

Cribl.Cloud supports only OIDC backchannel authentication, not front-channel.

This page is a guide for configuring SSO for a Cribl.Cloud deployment. For information about an on-prem installation, see SSO with Okta and OIDC (on-prem).

Set Up Fallback Access

Before you start configuring SSO, set up fallback access, so you are not locked out if issues with SSO occur.

In your Cribl.Cloud Organization, ensure that at least one Owner creates a local account, using an email domain that's separate from the corporate domain on which you're configuring SSO.

Bypassing SSO with Multiple Organizations

If you have SSO configured and you want to sign up for an additional Cribl.Cloud Organization, you need to bypass SSO. Otherwise, you will be forced to log into your existing Organization, because SSO does Home Realm Discovery and recognizes your email address.

In that case, edit your login URL and delete the word `identifier`. For example:

- Original URL: `https://login.cribl.cloud/u/login/identifier?state=<long_string_of_characters>`
- Edited URL: `https://login.cribl.cloud/u/login/?state=<long_string_of_characters>`

When you use this URL, instead of forcing you through SSO, Cribl.Cloud will ask for a username and password.

Create OIDC App Integration

To create your app integration:

1. In Okta, navigate to the **Applications** section and select **Create App Integration**.
2. Configure the app integration with the options below:
    - **Sign-in method**: `OIDC - OpenID Connect`

- **Application type**: `Web Application`

General Settings

3. Configure the app integration's **General Settings** with the options below:
   - **App integration name**: `Cribl.Cloud (OIDC)`
   - **Grant type**: Select `Authorization Code` and `Refresh Token`.
   - **Sign-in redirect URIs**:
     - `https://login.cribl.cloud/login/callback`
     - `https://manage.cribl.cloud/<organizationID>/organization/sso`
   - **Sign-out redirect URIs**: https://login.cribl.cloud/v2/logout

If your IDP is PingOne, you must also configure this (non-Okta) option:

- **Authentication options**: `Allow Client Secret`

Assignments

4. Configure the **Assignments** pane with the following options:
   - **Controlled access**: `Limited access to selected groups`
   - **Selected groups**: The groups you mapped in Configure Groups.
5. **Save** your application.
6. Now return to the **General** tab's **General Settings** section and in **Refresh token behavior**, select `Use persistent token`.

Sign On Tab

7. If you are *not* mapping Teams to IDP groups, you need to specify a groups claim filter. In the **OpenID Connect ID Token** section, select **Edit**, and set the **Groups claim filter** to: `groups` : `Starts with` : `Cribl`.
8. To obtain the **Issuer URL** that you'll need to provide to Cribl in the next section, change the value in the **Issuer** field from `Dynamic` to `Okta URL`.

This step concludes the setup procedure for Okta (or other IDP).

Submit Your App Info to Cribl

Next, provide Cribl with essential details about your application to implement the SSO setup on the Cribl side.

1. In Cribl Edge, on the top bar, select **Products**, and then select **Cribl**.
2. In the sidebar, select **Organization**, then **SSO Management**.
3. Above **Web Application Settings**, select **OIDC**.
4. The **Web Application Settings** are prefilled for you, so you only need to fill in the **Cribl Cloud SSO Settings** section with the following details from your IDP client configuration:
    - Client ID
    - Client Secret
    - Issuer URL. Copy the **Issuer** URL from the **Sign On** > **OpenID Connect ID Token** section of your Okta environment.

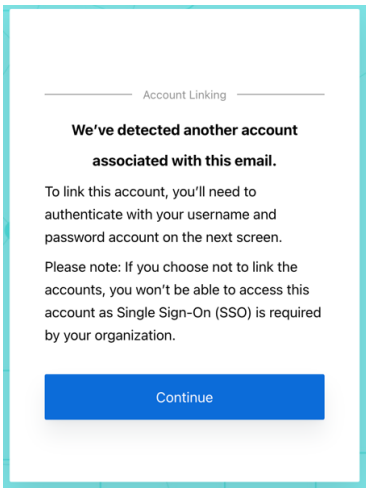
OIDC/Okta Chiclet Setup (Optional)

If you want to initiate login from your Okta instance with OIDC authentication configured, an Okta admin can configure an app integration as follows:

1. From Okta's left nav, select the **Applications** page.
2. Find the OIDC application created earlier in the OIDC/Okta Setup Example.
3. Select that application, and in the **General** tab's **General Settings** section, select **Edit**.
4. In the **Initiate login URI** field, enter `https://manage.cribl.cloud/login?connection=<organizationID>` (where `<organizationID>` is your Cribl.Cloud Organization's ID).
5. Confirm with **Save** to complete the chiclet.

Link Existing Users

If your Cribl.Cloud Organization has existing users who had been using a username and password to log in, upon first login with SSO, these users will see a prompt to link their identities. They should accept this prompt to ensure that their existing profile is linked with their SSO profile.



Prompt to link accounts

8.7.1.4. SSO With Microsoft Entra ID And SAML (Cribl.Cloud)

This page presents a walkthrough of setting up a SAML SSO, using Microsoft Entra ID as the example.

Set Up Fallback Access

Before you start configuring SSO, set up fallback access, so you are not locked out if issues with SSO occur.

In your Cribl.Cloud Organization, ensure that at least one Owner creates a local account, using an email domain that's separate from the corporate domain on which you're configuring SSO.

Bypassing SSO with Multiple Organizations

If you have SSO configured and you want to sign up for an additional Cribl.Cloud Organization, you need to bypass SSO. Otherwise, you will be forced to log into your existing Organization, because SSO does Home Realm Discovery and recognizes your email address.

In that case, edit your login URL and delete the word `identifier`. For example:

- Original URL: `https://login.cribl.cloud/u/login/identifier?state=<long_string_of_characters>`
- Edited URL: `https://login.cribl.cloud/u/login/?state=<long_string_of_characters>`

When you use this URL, instead of forcing you through SSO, Cribl.Cloud will ask for a username and password.

Create an Enterprise Application

From Microsoft Entra ID's left nav:

1. Select **Enterprise applications** and choose **All applications**, then **New application**.
2. Name your new app `Cribl.Cloud` (or any name you prefer).
3. Select **Integrate any other application you don't find in the gallery (Non-gallery)**.
4. Confirm with **Create**.

Assign Groups

Now, map your group from Cribl.Cloud to your SSO groups. From Microsoft Entra ID's left nav:

1. Select **Users and groups**, then select **Add user/group**.
2. Add the Cribl groups you created in Configure Groups. Do not configure Entra ID roles: access control will be handled by Cribl groups.

If Microsoft Entra ID is synchronized with your external Active Directory, you must set the groups claim: select `Groups assigned to the application` for association and set **Source attribute** to `sAMAccountName`.

Enable **Emit group name for cloud-only groups** to return the group names if it's defaulting to GUID or Object ID.

3. Click **Assign** after selecting Groups.

Configure Single Sign-On

From Microsoft Entra ID's left nav, select **Single sign-on**, then **SAML**, to open the **Basic SAML Configuration** page. Then, fill in the form with the information from Cribl.Cloud shown in the screenshot below:

1. In **Identifier (Entity ID)**, select **Add identifier** and enter the **Audience URI** value from the Cribl.Cloud SAML setup page.
2. In **Reply URL (Assertion Consumer Service URL)**, select **Add reply URL** and enter the two **Single Sign-on URL** values from Cribl.Cloud's SAML setup page.
3. Of these two URLs, identify the one with the `connection` query parameter, and select the check box to make it the **Default**.

SSO information for configuring Entra ID

Configure Attributes and Groups Claims

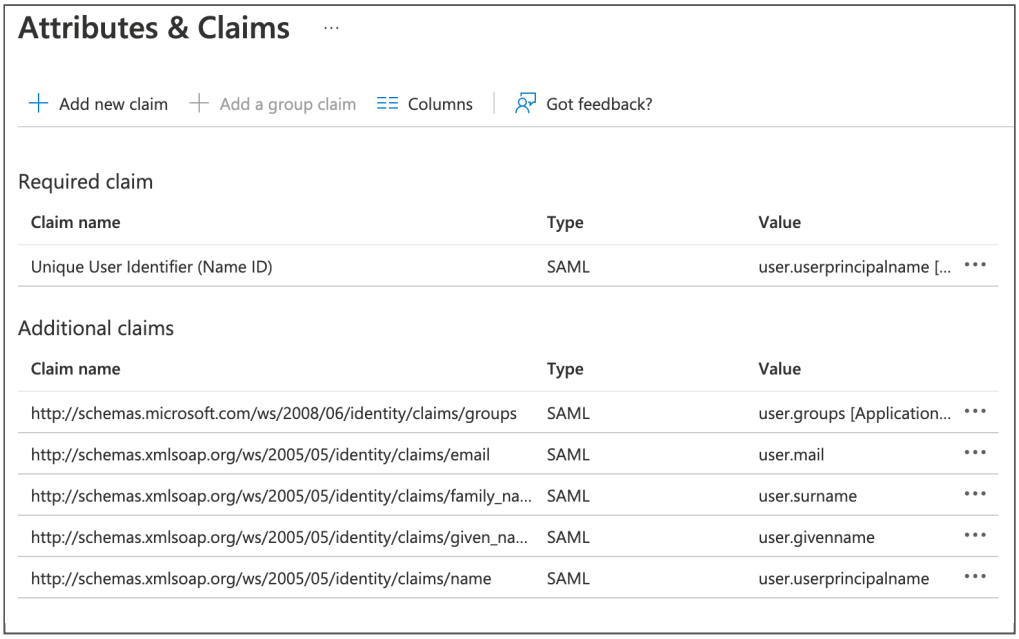
In Microsoft Entra ID, edit **User Attribute & Claims** as follows. Start with the claim names:

1. Change `surname` to `family_name`.

2. Change `emailaddress` to `email`.
3. Change `givenname` to `given_name`.

Next, add a group claim:

1. Select **Groups assigned to the application**.
2. Set the **Source attribute** to `Cloud-only group display names (Preview)`.
3. Accept the defaults for everything else, and save the new settings.



SAML Microsoft Entra ID Attribute

Submit Your App Info to Cribl

After you've created the SAML app integration in your IDP, provide Cribl with essential metadata about your application to implement SSO setup on the Cribl side.

1. In Cribl Edge, on the top bar, select **Products**, and then select **Cribl**.
2. In the sidebar, select **Organization**, then **SSO Management**.
3. Above **Web Application Settings**, select **SAML**.
4. Fill in the following fields with information from Entra ID:

Cribl.Cloud field	Entra ID
IDP Login/Logout URL	Login URL
IDP issuer	Microsoft Entra ID Identifier

Cribl.Cloud field	Entra ID
X.509 certificate (base64-encoded)	Certificate (Base64) under **SAML Certificates**

5. Select **Test Connection** to test the connection.
6. When you've verified the connection, select **Save** to complete your submission.

SAML/Entra ID Setup with My Apps Chiclet (Optional)

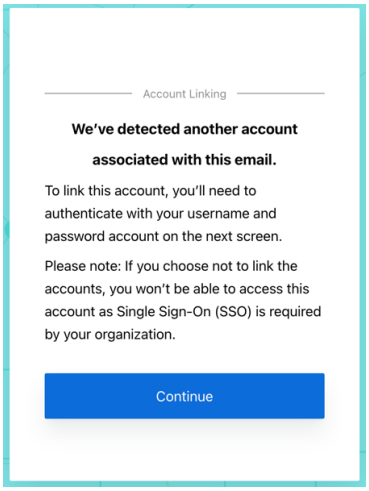
If you want to log into Cribl.Cloud via the Microsoft My Apps chiclet, complete the following procedure:

1. In Microsoft Entra ID, navigate to the enterprise application that you created to integrate SSO.
2. From the left nav, select **Single Sign-on**.
3. On the Enterprise Application's **Basic SAML Configurations** page, select **Edit**.
4. In the **Sign on URL (Optional)** section, enter the following URL: `https://manage.cribl.cloud/login?connection=<organizationID>` (where `<organizationID>` is your Cribl.Cloud Organization's ID).

You also need to allow self-service access to the Cribl App, or assign AD groups permissions to access the application.

Link Existing Users

If your Cribl.Cloud Organization has existing users who had been using a username and password to log in, upon first login with SSO, these users will see a prompt to link their identities. They should accept this prompt to ensure that their existing profile is linked with their SSO profile.



Prompt to link accounts

8.7.1.5. SSO With Okta And SAML (Cribl.Cloud)

This page presents a walkthrough of setting up a SAML SSO, using Okta as the example.

This page is a guide for configuring SSO for a Cribl.Cloud deployment. For information about an on-prem installation, see SSO with Okta and SAML (on-prem).

Set Up Fallback Access

Before you start configuring SSO, set up fallback access, so you are not locked out if issues with SSO occur.

In your Cribl.Cloud Organization, ensure that at least one Owner creates a local account, using an email domain that's separate from the corporate domain on which you're configuring SSO.

Bypassing SSO with Multiple Organizations

If you have SSO configured and you want to sign up for an additional Cribl.Cloud Organization, you need to bypass SSO. Otherwise, you will be forced to log into your existing Organization, because SSO does Home Realm Discovery and recognizes your email address.

In that case, edit your login URL and delete the word `identifier`. For example:

- Original URL: `https://login.cribl.cloud/u/login/identifier?state=<long_string_of_characters>`
- Edited URL: `https://login.cribl.cloud/u/login/?state=<long_string_of_characters>`

When you use this URL, instead of forcing you through SSO, Cribl.Cloud will ask for a username and password.

Create SAML 2.0 App Integration

To create your app integration:

1. In Okta, navigate to the **Applications** section of your Okta environment and select **Create App Integration**.
2. Create the app integration with **Sign-in method**: `SAML 2.0`.
3. Provide a name for your app and proceed with **Next**.

Configure SAML Settings

1. In the **Configure SAML** tab, in the **SAML Settings** section, you will use information you get from your Cribl.Cloud Organization. Configure the following options:
    - **Single sign-on URL** is the first of two URLs that your Cribl Organization's **SSO** > **SAML** tab lists under **Single Sign on URL**. For example: `https://login.cribl.cloud/login/callback?connection=<$organizationID>`
    - **Audience URI (SP Entity ID)** is located in your **SSO** > **SAML** tab under th same label. For example: `urn:auth0:cribl-cloud-prod:<$organizationID>`
    - **Application username**: `Email`

    The `nameidentifier` assertion in SAML responses must be the user's `Email`.
2. Select **Show Advanced Settings**. Navigate down and configure a single row of **Other Requestable SSO URLs**, as follows:
    - **URL** is required to test your connection. Get it from your Cribl.Cloud Organization's **SSO** > **SAML** tab, where it is the second **Single sign-on URL**. It will be in this format: `https://manage.cribl.cloud/api/assert`
    - **Index**: Set this to `0`.

Configure Attribute Statements

1. Configure **Attribute Statements** for these attributes, as shown below:

Name	Value
`email`	`user.email`
`given_name`	`user.firstName`
`family_name`	`user.lastName`

2. If you are *not* mapping Teams to IDP groups, you need to configure **Group Attribute Statements**:

Name	Filter
`groups`	Matches regex: `Cribl.*`

3. Save your app integration.

Submit Your App Info to Cribl

After you've created the SAML app integration in your IDP, provide Cribl with the essential metadata about your application to implement SSO setup on the Cribl side.

1. In Cribl Edge, on the top bar, select **Products**, and then select **Cribl**.
2. In the sidebar, select **Organization**, then **SSO Management**.
3. Above **Web Application Settings**, select **SAML**.
4. The **Web Application Settings** will be prefilled for you, and Cribl will also prefill the **SAML Assertion Mappings** based on the information you've registered with Cribl. So you only need to fill in the **SAML configuration** section with details from your IDP client configuration.
5. Return to your Okta environment to the **Sign On** tab and in the right pane, select **View SAML setup instructions**. Use the provided fields to fill in the information in Cribl.Cloud:

Cribl.Cloud field	Okta field
IDP Login/Logout URL	Identity Provider Single Sign-On URL
IDP issuer	Identity Provider Issuer
X.509 certificate (base64-encoded)	X.509 Certificate

SAML/Okta Chiclet Setup (Optional)

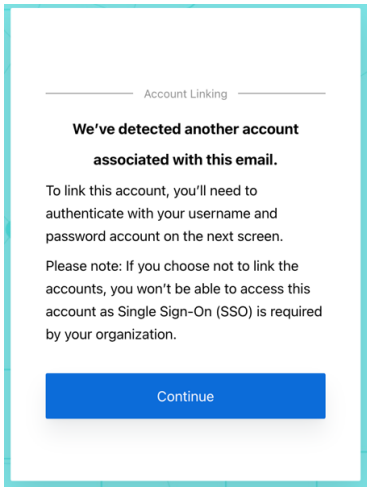
If you want to initiate login from your Okta instance with SAML authentication configured, an Okta admin can configure an app integration as follows:

1. From Okta's left nav, select the **Applications** page.
2. Select **Browse App Catalog**.
3. From the resulting catalog, use the search bar to find and select the `Bookmark App` application.
4. From that application's page, select **Add Integration**.
5. On the **General settings** page, enter an **Application label** that will identify this app as supporting Cribl.Cloud login. (`Cribl.Cloud` is a good choice, but the label is arbitrary.)
6. In the **URL** field, enter `https://manage.cribl.cloud/login?connection=<organizationID>` (where `<organizationID>` is your Cribl.Cloud Organization's ID).
7. Confirm with **Done**.
8. Select **Assign** and assign all of the Cribl.Cloud groups to the application.

9. The Cribl.Cloud chiclet should now be available for all users in the Cribl groups you've assigned.

Link Existing Users

If your Cribl.Cloud Organization has existing users who had been using a username and password to log in, upon first login with SSO, these users will see a prompt to link their identities. They should accept this prompt to ensure that their existing profile is linked with their SSO profile.



Prompt to link accounts

8.7.1.6. SSO Troubleshooting In Cribl.Cloud

Issues with enabling SSO typically result from incorrect or missing configuration. If your users are encountering problems logging in, make sure you have configured all the necessary fields (watch out for typos!):

- For SAML-based SSO, ensure that you have filled in the following fields correctly:
  - **IDP Login/Logout URL**
  - **IDP issuer**
  - **X.509 certificate (base64-encoded)**
- For OIDC-based SSO, verify the correctness of the following fields:
  - **Client ID**
  - **Client secret**
  - **Issuer URL**

Ensure all required users and groups are assigned to the application.

Make sure that the claims in your IDP are set up correctly. See our docs on Entra ID SAML and Okta OIDC.

If IDP-initiated logins are malfunctioning, you may see an error message: “IdP-Initiated login is not enabled”. In this case, refer to the chiclet setup docs for Okta/OIDC, Okta/SAML, or Microsoft Entra ID/SAML.

Troubleshooting Resources

Cribl University offers an SSO Integration – Cribl.Cloud – Okta Troubleshooting Criblet. 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. Cribl’s training is always free of charge.

Once logged in, check out other useful Troubleshooting Criblets and Advanced Troubleshooting short courses.

8.7.2. On-Prem SSO

8.7.2.1. SSO With Okta And OIDC (On-Prem)

Cribl Edge supports setting up SSO using OIDC to provide user authentication (login/password) and authorization (by mapping SSO users to Cribl Roles).

This page presents a walkthrough of setting up an OIDC SSO, using Okta as the example.

If you'd prefer to learn this information in the form of a video course, take a look at Cribl University's Troubleshooting Criblet on SSO Integration for Stream On-Prem - Okta. The course walks you through a configuration flow of SSO wth Okta and OIDC. (To follow the direct course link, first log into your Cribl University account - it's free!)

This page is a guide for configuring SSO for an on-prem installation. For Cribl.Cloud, see SSO with Okta and OIDC (Cribl.Cloud).

Create OIDC App Integration

To create your app integration:

1. In Okta, navigate to the **Applications** section and select **Create App Integration**.
2. Configure the app integration with the options below:
    - **Sign-in method**: `OIDC - OpenID Connect`
    - **Application type**: `Web Application`
3. Proceed with **Next**.

General Settings

4. Configure the app integration's **General Settings** with the options below:

Setting	Description
App integration name	You application name.
Logo	(Optional) Upload the Cribl logo. You can use a logo from the Cribl Press Kit.

Setting	Description
Sign-in redirect URIs	Your Leader base URL, and with `/api/v1/auth/authorization-code/callback` as the path.
Sign-out redirect URIs	(Optional) Append `/login` to the pre-filled path.

Assignments

5. In the **Assignments** pane, configure the access granted to Okta users:
    - To grant access to Cribl Edge to all Okta users, select **Allow everyone in your organization to access**.
    - To grant access to Cribl Edge to specific Okta groups, select **Limit access to selected groups**. Then, in the field below, add the groups you want to include.
      After you finish creating the app, if you need to add or remove groups, do that in the **Applications** > **Assignments** tab.
6. **Save** your application.

In the **Client Credentials** panel, note down **Client ID** and **Client Secret**. You will need them in the next step.

Submit Your App Info to Cribl

Next, provide Cribl with essential details about your application to implement the SSO setup on the Cribl side.

1. In Cribl Edge, in the sidebar, select **Settings**, then **Global**.
2. In **Access Management**, select **Authentication**.
3. From the **Type** dropdown, choose `OpenID Connect`.
4. In the **Provider name** dropdown, select `Okta`.
5. In the **Audience (Relying Party ID)** field, enter your Cribl Edge UI base URL.

If you have a Distributed deployment with a fallback Leader configured, modify the **Audience (Relying Party ID)** field to point to the load balancer instead of the Leader Node.

6. In the **Client ID** and **Client secret** fields, enter the respective values that you copied from the Okta **Client Credentials** in the previous step.
7. If your Cribl Edge is in Enterprise Distributed mode:
   In the **Scope** field, add the scope `groups` to the default space-separated list of scopes, so that it reads: `openid profile email groups`.

Get Okta App URLs

Next, get the `authentication`, `token`, `userinfo`, and `logout` URLs for your Okta app.

You can get them in JSON format from the `/.well-known/openid-configuration` endpoint: `https://<tenant>.okta.com/.well-known/openid-configuration` (`<tenant>` is your Okta tenant name). For example: `https://dev-12345678.okta.com/.well-known/openid-configuration`.

You can use a tool like jq to automatically extract the URLs from the endpoint's response with the following `curl` command:

```
curl -s https://<tenant>.okta.com/.well-known/openid-configuration | jq '. | {"a
```

Once you have the values, enter them in the respective fields in your Cribl Edge SSO configuration:

Cribl Edge field	Okta URL
Authentication URL	`authentication`
Token URL	`token`
User info URL	`userinfo`. Fill it in if you configured Okta to use groups.
Logout URL	`logout`. Fill it in if you want **Account** > **Log out** in Cribl Edge to log the user out globally. This means that when a user selects the **Accounts** > **Log out** link in Cribl Edge, they are logged out of both Cribl Edge and Okta.

Configure Response to Okta `/userinfo` Endpoint

An Okta tenant's user groups can be mastered either inside Okta, outside Okta, or both.

When Cribl Edge queries the `/userinfo` endpoint, Okta returns the appropriate groups membership of the user.

See the Okta documentation on dynamic allow lists and using Okta together with Active Directory.

To configure this behavior, you need to set **Groups claim type** correctly, depending on where user groups are mastered.

1. In Okta, go to your app and open the **Sign On** tab.
2. In the **OpenID Connect ID Token** panel, select **Edit**

Inside Okta | Outside Okta

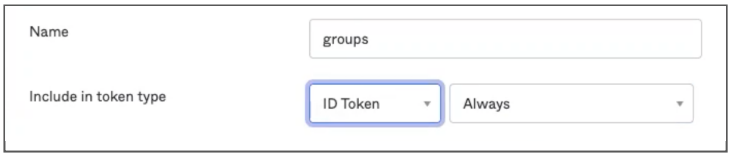
3. Set **Groups claim type** to `Filter`.
4. Choose **Matches regex** from the dropdown, and enter `.*` as the regex.

5. Click **Save**.

Configure ID Token to Include Groups Claim

For Okta to recognize your groups, you must configure the ID token to include your groups claim:

1. In Okta, open the **Security** > **API** page.
2. In the **Authorization Servers** tab, click the edit (pencil) button for the desired Authorization Server.
3. In the resulting page, select the **Claims** tab.
4. If your groups claim already exists, click the edit (pencil) button. Otherwise, click **Add Claim**.
5. In the **Include in token type** drop-downs, choose `ID Token` and `Always`, respectively.



Including the groups claim in the token ID

6. Configure the remaining settings in the way that suits your groups claim.
7. Click **Save** (or **Create** if you're adding the claim for the first time).

Map Okta Groups to Cribl Edge Roles

Mapping groups to Roles is possible only for Cribl Edge deployments that are in Distributed mode, with an Enterprise license. With a Standard license, all your external users will be imported to Cribl Edge in the `admin` role.

If you are running Cribl Edge in Single-instance mode, you cannot map Okta groups to Cribl Edge Roles, although you can still set up SSO with Okta.

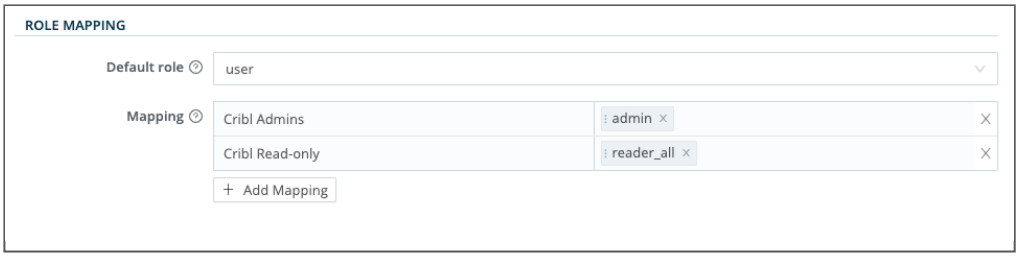
As you think through how best to map your Okta groups to Cribl Edge Roles, keep these principles in mind:

- An Okta group can map to more than one Cribl Edge Role.
- A Cribl Edge Role can map to more than one Okta group.
- If a user has multiple Roles, Cribl Edge applies the union of the most permissive levels of access.
- Cribl Edge automatically assigns the `default` Role to any user who has no mapped Roles.

For details on mapping your external identity provider's configured groups to corresponding Cribl Edge user access Roles, see External Groups and Roles.

You can assign a Cribl Edge Role to each Okta group name, and you can specify a `default` Role for users who are not in any groups.

1. In Cribl Edge, in the sidebar, select **Settings**, go to **Access Management** and then select **Authentication**.
2. Scroll down to **Role Mapping**.
   Cribl recommends that you set the `default` Role to `user`, meaning that this Role will be assigned to users who are not in any groups.
3. Add mappings as needed.
   The Okta group names in the left column are case-sensitive, and must match the values returned by Okta (those you saw earlier when configuring Okta and OIDC).



Example Role mapping

Verify that SSO with Okta Is Working

1. Log out of Cribl Edge, and verify that Okta is now an option on the login page.
2. Select **Log in with Okta**.
3. You should be redirected to Okta to authenticate yourself.
4. The OpenID connect flow should complete the authentication process.

Getting Temporary Access Credentials for AWS S3 Buckets

You can use your SSO/OIDC IDP to issue temporary access credentials so your on-prem Edge Node can access AWS S3 buckets.

Set the `AWS_WEB_IDENTITY_TOKEN_FILE` environment variable. This variable defines a path to a file that contains the OAuth/OIDC provided by the SSO IDP. You'll also need to define `AWS_ROLE_ARN` and `AWS_ROLE_SESSION_NAME`.

You can use curl/Postman to make the required API calls.

Ensure Fallback Access

To ensure fallback access for local users if SSO authentication fails, enable local authentication.

1. In the sidebar, select **Settings**, then **Global**.
2. In **Access Management**, select **Authentication**.
3. Toggle **Allow local auth** to **Yes** to allow users to log in using Cribl local authentication. This enables an extra button called `Log in with local user` on the login page.

To prevent lockout, Cribl strongly recommends enabling **Allow local auth** until you're certain that external auth is working as intended. If you do get locked out, see Manual Password Replacement.

OIDC/Okta Chiclet Setup (Optional)

If you want to initiate login from an Okta instance on which you have configured OIDC authentication, an Okta admin can configure an app integration as follows:

1. From Okta's left nav, select the **Applications** page.
2. Find the OIDC application created earlier in the OIDC/Okta Setup Example.
3. Select that application, and in the **General** tab's **General Settings** section, select **Edit**.
4. In the **Initiate login URI** field, enter the `<host>:<port>` of your Cribl Leader Node.
5. Confirm with **Save** to complete the chiclet.

8.7.2.2. SSO With Okta And SAML (On-Prem)

Cribl Edge supports setting up SSO using SAML to provide user authentication (login/password) and authorization (by mapping SSO users to Cribl Roles).

This page presents a walkthrough of setting up a SAML SSO, using Okta as the example.

This page is a guide for configuring SSO for an on-prem installation. For Cribl.Cloud, see SSO with Okta and SAML (Cribl.Cloud).

Create SAML 2.0 App Integration

To create your app integration:

1. In Okta, navigate to the **Applications** section and select **Create App Integration**.
2. In **Sign-in method**, select `SAML 2.0`.
3. Proceed with **Next**.

General Settings

4. Configure the app integration's **General Settings** with the options below:

Setting	Description
App integration name	You application name.
Logo	(Optional) Upload the Cribl logo. You can use a logo from the Cribl Press Kit.

SAML Settings

5. In the **Configure SAML** tab configure the following options:

Setting	Description
Single sign-on URL	Cribl Edge **Sign-on callback URL**.
Audience URI (SP Entity ID)	Cribl Edge **Audience (SP entity ID)**.

Setting	Description
Application username	A plain username, an email, or a custom username. In the SAML assertion's `subject` statement, this is the value for `NameID`. By default, this value will be the username in Cribl Edge. Alternatively, you can set a custom attribute statement in Okta, then set the **Username field** in Cribl Edge to use that instead.

6. Define custom **Attribute Statements** that Okta will insert in the SAML assertions shared with Cribl Edge. This applies only when creating a custom **Application username**, as described in the previous step.
7. Configure **Group Attribute Statements**. Similar to the previous step, except that here, Cribl Edge supports creating a custom attribute whose value is one or more Okta groups that will populate Cribl Edge's **Group name field**.
8. To check the SAML assertion in XML form, click **Preview the SAML Assertion**.
9. Skip thought the **Feedback** pane and **Save** your application.

Submit Your App Info to Cribl

Next, provide Cribl with essential details about your application to implement the SSO setup on the Cribl side.

1. In Cribl Edge, in the sidebar, select **Settings**, then **Global**.
2. In **Access Management**, select **Authentication**.
3. From the **Type** dropdown, choose `SAML 2.0`.
4. In the **Audience (SP entity ID)** field, enter the base URL of your Cribl Edge instance, for example, `https://yourDomain.com:9000`. Do **not** append a trailing slash.

If you have a Distributed deployment with a fallback Leader configured, modify the **Audience (SP entity ID)** field to point to the load balancer instead of the Leader Node.

5. Return to your Okta environment to the **Sign On** tab and in the right pane, select **View SAML setup instructions**. Use the provided fields to fill in the information in Cribl Edge:

Cribl Edge field	Okta field
Single sign-on (SSO) URL	Identity Provider Single Sign-On URL
Single logout (SLO) URL	Identity Provider Single Logout URL
Issuer (IDP entity ID)	Identity Provider Issuer
Response validation certificate	X.509 Certificate

Map Okta Groups to Cribl Edge Roles

Mapping groups to Roles is possible only for Cribl Edge deployments that are in Distributed mode, with an Enterprise license. With a Standard license, all your external users will be imported to Cribl Edge in the `admin` role.

If you are running Cribl Edge in Single-instance mode, you cannot map Okta groups to Cribl Edge Roles, although you can still set up SSO with Okta.

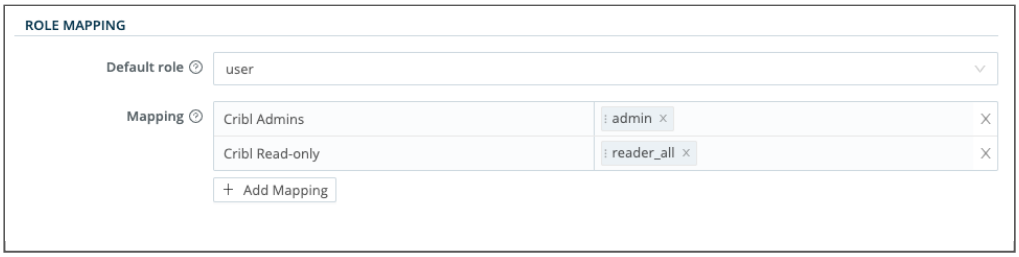
As you think through how best to map your Okta groups to Cribl Edge Roles, keep these principles in mind:

- An Okta group can map to more than one Cribl Edge Role.
- A Cribl Edge Role can map to more than one Okta group.
- If a user has multiple Roles, Cribl Edge applies the union of the most permissive levels of access.
- Cribl Edge automatically assigns the `default` Role to any user who has no mapped Roles.

For details on mapping your external identity provider's configured groups to corresponding Cribl Edge user access Roles, see External Groups and Roles.

You can assign a Cribl Edge Role to each Okta group name, and you can specify a `default` Role for users who are not in any groups.

1. In Cribl Edge, in the sidebar, select **Settings**, go to **Access Management** and then select **Authentication**.
2. Scroll down to **Role Mapping**.
   Cribl recommends that you set the `default` Role to `user`, meaning that this Role will be assigned to users who are not in any groups.
3. Add mappings as needed.
   The Okta group names in the left column are case-sensitive, and must match the values returned by Okta (those you saw earlier when configuring Okta and SAML).



Example Role mapping

Verify that SSO with Okta Is Working

1. Log out of Cribl Edge, and verify that Okta is now an option on the login page.
2. Select **Log in with Okta**.
3. You should be redirected to Okta to authenticate yourself.
4. The SAML connect flow should complete the authentication process.

Getting Temporary Access Credentials for AWS S3 Buckets

You can use your SSO/SAML IDP to issue temporary access credentials so your on-prem Edge Node can access AWS S3 buckets.

Call the `AssumeRoleWithSAML` API endpoint. It will return the STS access, secret, and session tokens. These can be written into the `~/.aws/` credentials file, which Cribl Edge will pick up because it uses the native AWS SDK.

You can set up multiple S3 Sources with different credentials. Cribl Edge relies on the AWS SDK for authentication support, and the SDK evaluates credentials in the following order:

1. Loaded from AWS Identity and Access Management (IAM) roles for Amazon EC2.
2. Loaded from the shared credentials file (~/.aws/credentials).
3. Loaded from environment variables.
4. Loaded from a JSON file on disk.
5. Other credential-provider classes provided by the JavaScript SDK.

To enable the use of multiple Sources, set the S3 Source **Authentication method** to **Auto**.

See the AWS SDK documentation and AWS CLI documentation for further information on setting credentials.

Ensure Fallback Access

To ensure fallback access for local users if SSO authentication fails, enable local authentication.

1. In the sidebar, select **Settings**, then **Global**.
2. In **Access Management**, select **Authentication**.
3. Toggle **Allow local auth** to **Yes** to allow users to log in using Cribl local authentication. This enables an extra button called `Log in with local user` on the login page.

To prevent lockout, Cribl strongly recommends enabling **Allow local auth** until you're certain that external auth is working as intended. If you do get locked out, see Manual Password Replacement.

SAML/Okta Chiclet Setup (Optional)

If you want to initiate login from an Okta instance on which you have configured SAML authentication, an Okta admin can configure an app integration as follows:

1. From Okta's left nav, select the **Applications** page.
2. Select **Browse App Catalog**.
3. From the resulting catalog, use the search bar to find and select the `Bookmark App` application.
4. From that application's page, select **Add Integration**.
5. On the **General settings** page, enter an **Application label** that will identify this app as supporting Cribl login. (`Cribl` is a good choice, but the label is arbitrary.)
6. In the **URL** field, enter the `<host>:<port>` of your Cribl Leader Node.
7. Confirm with **Done**.
8. Select **Assign** and assign all of the Cribl groups to the application.
9. The `Cribl` chiclet should now be available for all users in the Cribl groups you've assigned.

8.7.2.3. SSO With Microsoft Entra ID And SAML (On-Prem)

Cribl Edge supports setting up SSO using SAML to provide user authentication (login/password) and authorization (by mapping SSO users to Cribl Roles).

This page presents a walkthrough of setting up a SAML SSO, using Microsoft Entra ID (formerly Azure AD) as the example.

This page is a guide for configuring SSO for an on-prem installation. For Cribl.Cloud, see SSO with Microsoft Entra ID and SAML (Cribl.Cloud).

Create an Enterprise Application

To create your app integration:

1. In Microsoft Entra ID's top nav, select **Add**, then **Enterprise application**, then **Create your own application**.
2. Name your new app `Cribl` (or any name you prefer).
3. Select **Integrate any other application you don't find in the gallery (Non-gallery)**.
4. Confirm with **Create**.

Submit Your App Info to Cribl

Next, provide Cribl with essential details about your application to implement the SSO setup on the Cribl side.

1. In Cribl Edge, in the sidebar, select **Settings**, then **Global**.
2. In **Access Management**, select **Authentication**.
3. From the **Type** dropdown, choose `SAML 2.0`.
4. In the **Audience (SP entity ID)** field, enter the base URL of your Cribl Edge instance, for example, `https://yourDomain.com:9000`. Do **not** append a trailing slash.

If you have a Distributed deployment with a fallback Leader configured, modify the **Audience (SP entity ID)** field to point to the load balancer instead of the Leader Node.

5. Then fill in remaining information from the Entra ID's **SAML-based Sign-on** page:

Cribl Edge field	Entra ID field
Issuer (IDP entity ID)	Microsoft Entra Identifier
Single sign-on (SSO) URL	Login URL
Single logout (SLO) URL	Logout URL
Response validation certificate	Certificate (Base64) under **SAML Certificates**
Group name field	`http://schemas.microsoft.com/ws/2008/06/identity/claims/groups`

5. Select **Save** to complete your submission.

Configure Single Sign-On

From Microsoft Entra ID's left nav, under **Manage**, select **Single sign-on**, then **SAML**, to open the **SAML-based Sign-on** page. Then, edit the fields below with the information from the Cribl **Authentication** page:

1. In **Identifier (Entity ID)**, select **Add identifier** and enter the **Audience URI** value from the Cribl **Authentication** page.
2. In **Reply URL (Assertion Consumer Service URL)**, select **Add reply URL** and enter the **Sign-on callback URL** value from the Cribl **Authentication** page.
3. In **Logout Url (Optional)**, enter the **Logout callback URL** value from the Cribl **Authentication** page.

Map Entra ID Groups to Cribl Edge Roles

Mapping groups to Roles is possible only for Cribl Edge deployments that are in Distributed mode, with an Enterprise license. With a Standard license, all your external users will be imported to Cribl Edge in the `admin` role.

If you are running Cribl Edge in Single-instance mode, you cannot map Entra ID groups to Cribl Edge Roles, although you can still set up SSO with Entra ID.

As you think through how best to map your Entra ID groups to Cribl Edge Roles, keep these principles in mind:

- An Entra ID group can map to more than one Cribl Edge Role.
- A Cribl Edge Role can map to more than one Entra ID group.
- If a user has multiple Roles, Cribl Edge applies the union of the most permissive levels of access.

- Cribl Edge automatically assigns the `default` Role to any user who has no mapped Roles.

For details on mapping your external identity provider's configured groups to corresponding Cribl Edge user access Roles, see External Groups and Roles.

You can assign a Cribl Edge Role to each Entra ID group name, and you can specify a `default` Role for users who are not in any groups.

1. In Cribl Edge, in the sidebar, select **Settings**, go to **Access Management** and then select **Authentication**.
2. Scroll down to **Role Mapping**.
   Cribl recommends that you set the `default` Role to `user`, meaning that this Role will be assigned to users who are not in any groups.
3. Add mappings as needed.
   The Entra ID group names in the left column are case-sensitive, and must match the values returned by Entra ID .

Assign Groups

Now, map your Cribl groups to your Entra ID groups. From Microsoft Entra ID's left nav:

1. Select **Manage**, then **Users and groups**, then select **Add user/group**.
2. Add the Cribl groups you created above

You need to synchronize Microsoft Entra ID with your on-premises Active Directory to be able to configure returning `sAMAccountName` for group names. Otherwise, Entra ID will return only GUIDs.

For the groups claim configuration, select `Groups assigned to the application` for association and set **Source attribute** to `sAMAccountName`. Enable **Emit group name for cloud-only groups** to also return the `sAMAccountName` attribute for cloud-only groups.

Verify that SSO with Entra ID Is Working

1. Log out of Cribl Edge, and verify that Entra ID is now an option on the login page.
2. Select **Log in with Saml 2.0**.
3. You should be redirected to Entra ID to authenticate yourself.
4. The SAML connect flow should complete the authentication process.

Ensure Fallback Access

To ensure fallback access for local users if SSO authentication fails, enable local authentication.

1. In the sidebar, select **Settings**, then **Global**.
2. In **Access Management**, select **Authentication**.
3. Toggle **Allow local auth** to **Yes** to allow users to log in using Cribl local authentication. This enables an extra button called `Log in with local user` on the login page.

To prevent lockout, Cribl strongly recommends enabling **Allow local auth** until you're certain that external auth is working as intended. If you do get locked out, see Manual Password Replacement.

8.7.2.4. SSO With Microsoft Entra ID And OIDC (On-Prem)

Cribl Edge supports setting up SSO using OIDC to provide user authentication (login/password) and authorization (by mapping SSO users to Cribl Roles).

This page presents a walkthrough of setting up an OIDC SSO, using Microsoft Entra ID (formerly Azure AD) as the example.

Register Your Microsoft Entra ID App

1. Open the Microsoft Entra ID portal and log in.
2. In the left nav's **Manage** section, select **App registrations**.
3. Select **New registration** and provide a name for your app.
4. In **Supported account types**, select **Accounts in this organizational directory only (Default Directory only - Single tenant)**.
5. In **Redirect URI (optional)**, select **Web** and provide the appropriate callback URL for your own Cribl Edge Leader instance, for example: `https://yourDomain.com:9000/api/v1/auth/authorization-code/callback`

Create and Copy a Client Secret

Next, create a client secret:

1. From your app's left nav, select **Manage** > **Certificate & secrets**. Then select **New client secret**.
2. Add a new client secret with a descriptive name, and an expiration timeframe.
3. Confirm with **Add**.
4. Immediately copy the **Value** and **Secret ID** from the resulting page. You'll need to paste the **Value** into Cribl Stream's **Authentication** > **Client secret** field below.

This is the only time the secret is shown! Make sure you copy it while it's visible. (If you missed your chance, you can start over by creating a new secret.)

Configure Token and Claims

Here, you'll add the groups claim to the OIDC ID token.

1. From your app's left nav, select **Manage** > **Token configuration**, then select **Add groups claim**.
2. Under **Select group types to include in Access, ID, and SAML tokens**, select **Groups assigned to the application**.
3. Confirm with **Add**.

You need to synchronize Microsoft Entra ID with your on-premises Active Directory to be able to configure returning `sAMAccountName` for group names. Otherwise, Entra ID will return only GUIDs.

In the **Group claims** modal check **Emit group name for cloud-only groups** to ensure that `sAMAccountName` is added to the **Group** attribute.

Submit Your App Info to Cribl

Next, provide Cribl with essential details about your application to implement the SSO setup on the Cribl side.

1. In Cribl Edge, navigate to **Settings** > **Global**.
2. In **Access Management**, select **Authentication**.
3. From the **Type** dropdown, choose `OpenID Connect`.
4. In **Provider name**, enter an arbitrary identifier for this integration.
5. In the **Audience (Relying Party ID)** field, enter your Cribl Edge Leader's base URL. Do **not** append a trailing slash.

   If you have a Distributed deployment with a fallback Leader configured, modify the **Audience (Relying Party ID)** field to point to the load balancer instead of the Leader Node.
6. Fill in the remaining fields:

Cribl Edge field	Entra ID field
Client ID	**Application (client) ID** from Entra ID app overview.
Client secret	The secret value your copied earlier.
Authentication URL	**OAuth 2.0 authorization endpoint (v2)** from Entra ID app overview > **Endpoints**.
Token URL	**OAuth 2.0 token endpoint (v2)** from Entra ID app overview > **Endpoints**.

7. If your Cribl Edge is in Enterprise Distributed mode:
   In the **Scope** field, add the scope `groups` to the default space-separated list of scopes, so that it reads: `openid profile email groups`.

8. Change the **Filter type** to `User info filter`.

v1 Endpoints

You can alternatively use v1 endpoints instead of v2. In that case, replace the values for **Authentication URL** and **Token URL** with their v1 equivalents from Entra ID. When using v1 endpoints, you also need to modify the default value for **User identifier**.

In v1 only the `name` field is included in the token by default, so an acceptable entry here might be: `` `${unique_name || upn || username || name}` ``. You can check the token fields returned by enabling debug-level logging on Cribl Stream's `auth:sso` channel.

Map Microsoft Entra ID Groups to Cribl Stream Roles

Next, map your Microsoft Entra ID groups to Cribl Stream Roles.

Unless you synchronize Entra ID with your on-premises Active Directory, the group names will appear as GUIDs. You can view the groups and their GUIDs on the Entra ID **Groups** page.

To map the groups:

1. In Entra ID, in the left nav's **Manage** section, select **Groups**.
2. Find the group you want to configure and copy its **Object Id**.
3. In Cribl Edge, go to **Global Settings**.
4. Under **Access Management**, select **Authentication**.
5. Under **Role Mapping**, under **Mapping**, paste the group's object ID.
6. In the right column, select the Cribl Role you want to assign to this group.
7. Repeat this for all groups you want to configure.
8. Confirm with **Save**.

Ensure Fallback Access

To ensure fallback access for local users if SSO authentication fails, enable local authentication.

1. In the sidebar, select **Settings**, then **Global**.
2. In **Access Management**, select **Authentication**.
3. Toggle **Allow local auth** to **Yes** to allow users to log in using Cribl local authentication. This enables an extra button called `Log in with local user` on the login page.

To prevent lockout, Cribl strongly recommends enabling **Allow local auth** until you're certain that external auth is working as intended. If you do get locked out, see Manual Password Replacement.

9. Securing

9.1. Secure Your Cribl.Cloud Deployment

Cribl.Cloud is SOC 2 Type II–certified and GDPR-compliant, simplifying compliance with industry standards and regulatory requirements. 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 Managing

Cribl.Cloud.

Set Up Your Preferred Authentication Method

Cribl.Cloud with certain plan tiers offers a range of secure and convenient login options to fit your organization's needs. Options include:

- 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.

Secure Data in Motion

Cribl.Cloud provides built-in encryption for data at rest, ensuring that your information remains secure within its infrastructure. For an extra layer of protection, you can enrypt individual fields within events. 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.

Configure a Key Management Service (KMS)

Cribl.Cloud with certain plan tiers 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.

9.2. Secure Your On-Prem/Hybrid Deployment

This document outlines essential security considerations to safeguard your on-premises/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.

Locate and Access Your Cribl Secrets

When you install Cribl Edge, the following security files are created:

- `cribl.secret` – This file stores a unique cryptographic key used for authentication, password encryption, and key encryption. It's essential for securing your Cribl Edge deployment when authenticating with integrated services.
- `keys.json` – This file contains cryptographic keys used for internal operations within Cribl Edge.

These files are stored in:

Single-instance deployment – both files are located in `$CRIBL_HOME/local/cribl/auth/`.

Distributed deployment – both files reside on the Leader Node, in `$CRIBL_HOME/groups/<group-name>/local/cribl/auth/`.

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 and SAML (On-Prem)
- SSO with Microsoft Entra ID and OIDC (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 SSL.

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 Worker 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.
- 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 certain 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 find out more about secrets, see The cribl.secret File.
- To access secrets, see Accessing Secrets.

- To add a new secret, see Add New Secret..
- For secret types, see Secret Type.

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.

9.3. 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.

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** to `No`.
5. Select **Save**.

Connect to the Leader Securely

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

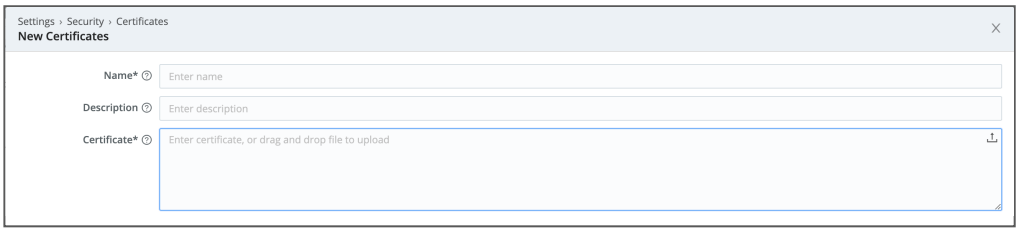
To set up secure communication via the UI, you configure first the Fleet, then the Edge Node, then the Leader.

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.

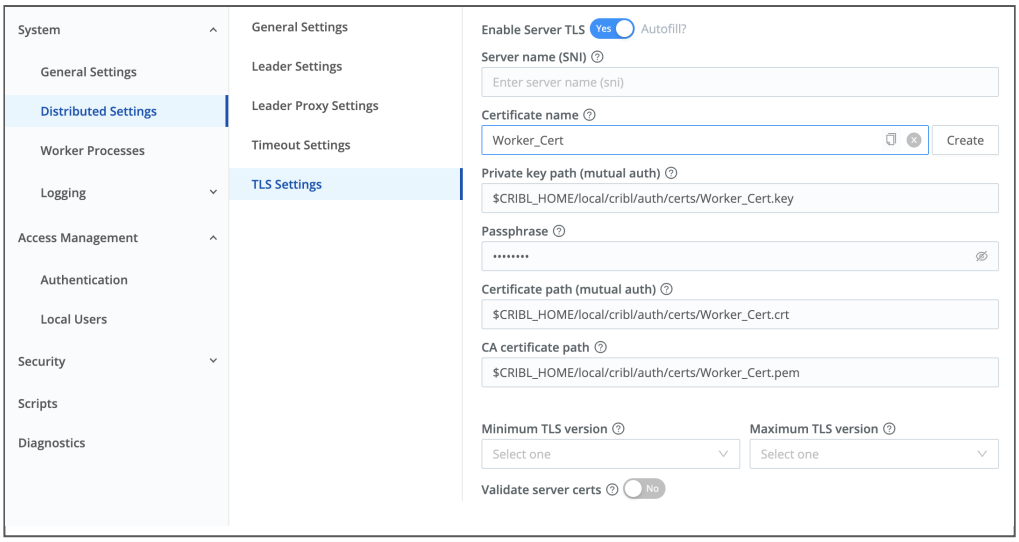


Fleet-level certificates are configured like the Leader's certificate

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** to `Yes`.
   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** to `Yes`.
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 prepend `https://` to 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
```

After you've enabled TLS on the Leader, generating bootstrap scripts to add or update Edge Nodes will automatically prepend `https://` to 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_MASTER_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. We can secure it using the self-signed certificate we created in Configure SSL Certificates:

1. Navigate to **Settings** > **Global** > **General Settings** > **API Server Settings** > **TLS**.
2. Slide the toggle to **Enabled**.
3. From the **Certificate Name** drop-down, select a cert 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. E.g., to get back to the Settings page you just configured, you'll now need to use `https://<hostname>:<port>/settings/system)`.

9.3.1. How To Secure The Auth Token For The Leader Node

For new installations, the Cribl Edge Leader 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 before v4.5.0, the default token was `criblmaster`. This default auth token does not change automatically if you upgrade your deployment to v4.5.0 or later.

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.

Before 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 Cribl Edge 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-
```

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**.

9.3.2. 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.

TLS Settings and Traffic Types

This table shows TLS client/server pairs, and encryption defaults, per traffic type.

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
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

Traffic Type	TLS Client	TLS Server	Encryption	Cert Auth	CN* Check
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 this environment variable for each Edge Node (for example, in its systemd unit file): `NODE_EXTRA_CA_CERTS=/path/to/file_with_certs.pem`. For details, see the nodejs docs.
- **Configure Cribl Edge to accept all TLS certificates, regardless of their validity**
  Set this environment variable: `NODE_TLS_REJECT_UNAUTHORIZED=0` – for details, see the nodejs docs.

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.

9.3.3. Securing Cribl.Cloud With TLS And Mutual TLS

Cribl.Cloud prioritizes secure data transfer with pre-enabled TLS on many sources. This guide details configuring TLS and enabling mutual TLS (mTLS) for additional sources.

For detailed instructions on configuring certificates to send Palo Alto logs to Cribl.Cloud via Syslog, refer to the guide: Palo Alto Syslog to Cribl.Cloud.

TLS in Cribl.Cloud

TLS encryption is pre-enabled on several Sources in Cribl.Cloud, as indicated on the **Data Sources** tab of the Cribl.Cloud portal. All TLS is terminated by individual Nodes.

If you enable TLS for additional Sources, the UI will autofill these configuration settings:

- **Private key path:** `<path-to-certificates>/$CRIBL_CLOUD_KEY`
- **Certificate path**: `<path-to-certificates>/$CRIBL_CLOUD_CRT`
- **Minimum TLS version**: `TLSv1.2` – the default; you can select a different version from the drop-down

The Cribl.Cloud Leader connections on port `4200/tcp` do not currently support mutual TLS (mTLS).

Enable mTLS Authentication on Cribl.Cloud

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. You 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 any valid PEM-formatted content:

```
-----BEGIN CERTIFICATE-----
CERTIFICATE CONTENT
-----END CERTIFICATE-----
```

The certificate and key are required only for UI validation and are not used otherwise.

4. Populate the **Private key** with the key in PEM format:

```
-----BEGIN RSA PRIVATE KEY-----
HIDDEN PRIVATE KEY
-----END RSA PRIVATE KEY-----
```

5. In the **CA certificate** field, enter your PEM-formatted certificate and save. This will generate the CA certificate path.
6. Edit the certificate again to view the certificate path and copy it.
7. Go to the Source where you want to enable mTLS and paste the path in the **CA certificate path** field.
8. Save, commit, and deploy to finish the process.

9.3.4. 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).

Limitations on mTLS Authentication

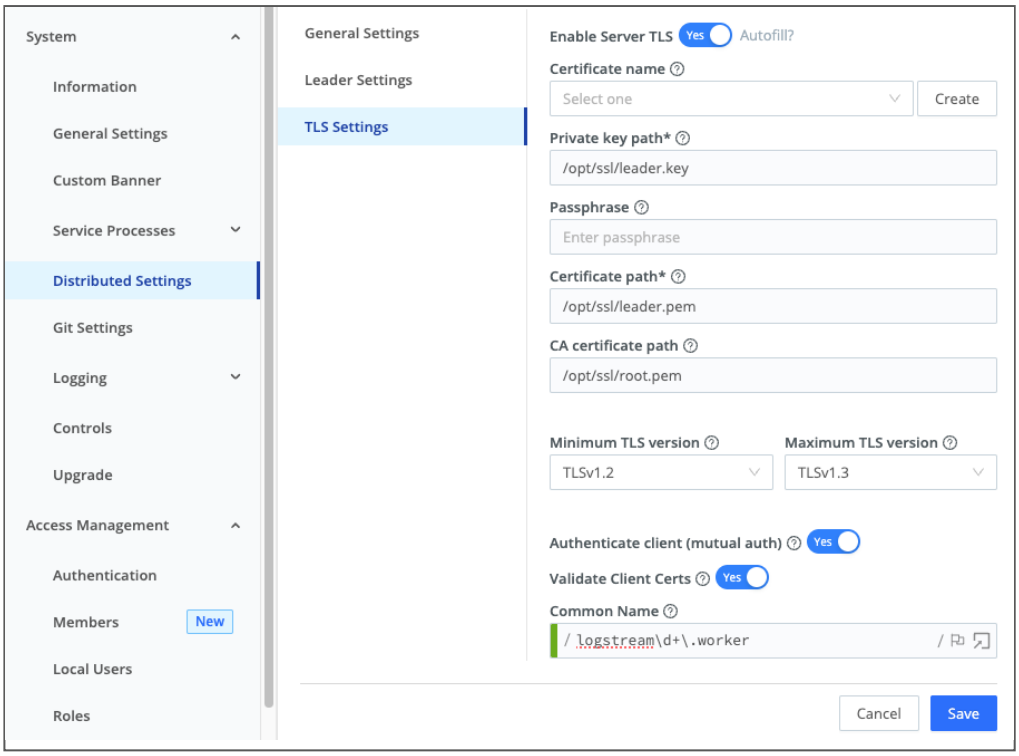
When configuring mTLS authentication on Edge Nodes, make sure you place your certificates into a separate directory outside of `$CRIBL_HOME`. If you place the certificates inside `$CRIBL_HOME`, they'll be removed when the next config bundle is deployed from the Leader.

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
```

Using Environment Variables

You can set up mTLS authentication by configuring this environment variable, using the format shown in this example:

```
CRIBL_DIST_Master_URL="tls://<authToken>@leader.cribl:4200?tls.privKeyPath=/path/to/
```

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.

9.3.5. Secure Leader Using An Existing TLS/SSL Certificate And Key

This page outlines how to protect Cribl Edge Leader to Edge Nodes communications, using an existing TLS/SSL certificate and key.

Cribl Edge expects certificates and keys to be formatted in privacy-enhanced mail (`.pem`) format. To generate a self-signed certificate and corresponding key, see Securing Cribl Edge.

Import Certificate and Key

To use your certificate and key to prepare secure communications between Workers/Edge Nodes and the Leader:

1. Navigate to the Leader's **Settings** > **Globals** > **Security** > **Certificates** > **New Certificates** modal.
2. Open your TLS/SSL certificate file. (The self-signed certificate example in Configure TLS for API and UI Access used the placeholder name `myCert.pem`.)
3. Copy the file's contents to your clipboard.
4. Paste the file's contents into the modal's **Certificate** field.

You can skip the preceding three steps: Just drag/drop your `.pem` file from your filesystem into the **Certificate** field, or upload it using the button at the field's upper right.

5. Open your private key file. (The self-signed certificate example in Configure TLS for API and UI Access used the placeholder name `myKey.pem`.)
6. Copy the file's contents to your clipboard.
7. Paste the clipboard contents into the same Leader modal's **Private key** field.

Here again, you can skip the preceding three steps by dragging/dropping or uploading the `.pem` file from your filesystem.

8. Fill in the **Name** and **Description** fields.
9. If you've uploaded a self-signed certificate, just **Save** it now.
10. If your private key is encrypted, fill in the modal's **Passphrase** with the corresponding key. (You can paste the key's contents, or you can drag/drop or upload the key file.)
11. If you're uploading a certificate signed by an external certificate authority – For example, a downloaded Splunk Cloud certificate – import the chain into the **CA certificate** field before saving the certifcate. For

details, see Obtain the Certificate Chain (TLS/SSL).

9.3.6. Configure TLS For API And UI Access

To safeguard inbound communications to your Edge Nodes, configure Transport Layer Security (TLS) on the Leader. This will enable secure access to the Cribl Edge API and UI.

In Cribl Edge v.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 to a pre-4.1 version, see Restore Unencrypted Private Keys below.

You can use your own certificates and private keys, or you can generate a pair with OpenSSL, as shown here:

```
openssl req -nodes -new -x509 -newkey rsa:2048 -keyout myKey.pem -out myCert.pem -days 420
```

This example command will generate both a self-signed cert `myCert.pem` (certified for 420 days), and an unencrypted, 2048-bit RSA private key `myKey.pem`. (Change the filename arguments to modify these placeholder names.)

As indicated by these examples, Cribl Edge expects certificates and keys to be formatted in privacy-enhanced mail (`.pem`) format.

In the Cribl Edge UI, you configure the certificate and its associated private key differently depending on your deployment type:

- For Single-instance deployments, navigate to **Settings** > **Globals** > **Security** > **Certificates** (Edge, Stream).
- For Distributed deployments, navigate to a Fleet and select **Fleet Settings** > **Security** > **Certificates** (Edge, Stream).

Alternatively, in the `local/cribl.yml` file, you can edit the `api` section to directly set the `privKeyPath` and `certPath` attributes. For example:

cribl.yml

```
api:
  host: 0.0.0.0
  port: 9000
  disabled : false
  ssl:
    disabled: false
    privKeyPath: /path/to/myKey.pem
    certPath: /path/to/myCert.pem
...
```

See Secure Leader/Edge Nodes Communication for details about using this certificate and key to secure communications on, and among, your Cribl Edge Leader and Edge Nodes.

Restore Unencrypted Private Keys

If you need to roll back from Cribl Edge v.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 v.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.

9.3.7. Secure Cribl Edge Sources And Destinations With Certificates

You can use certificates and custom HTTP headers to authenticate Cribl Edge with external Sources and Destinations, enhancing the security of your data pipelines.

Setting Authentication on Sources/Destinations

You can use certificates to authenticate Cribl Edge to external data senders and receivers. In the Cribl Edge UI, you configure the certificate and its associated private key differently depending on your deployment type:

- For Single-instance deployments, navigate to **Settings** > **Security** > **Certificates** (Edge, Stream). Next, select **Add Certificate**.
- For Distributed deployments, navigate to a Fleet and select **Fleet Settings** > **Security** > **Certificates** (Edge, Stream). Next, select **Add Certificate**.
- On Cribl.Cloud, navigate to **Settings** > **Global** > **Security** > **Certificates**.

As an alternative to the preconfiguration in steps 1–4 below, you can import a certificate on the fly, using the Source's or Destination's **Create** button.

In the **New Cerfificates** modal:

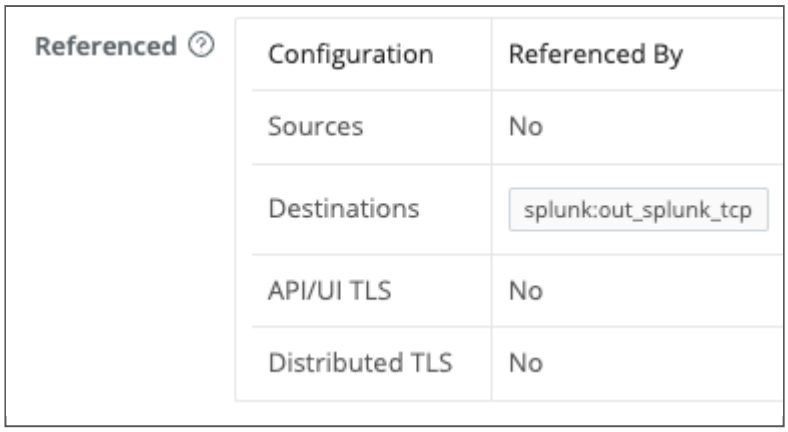
1. Paste or upload `.pem` files, according to the instructions in Import Certificate and Key.
2. Supply a **Passphrase** and/or **CA certificate**, if required by your integration partner's certificate.
3. Click **Save**.
4. Open your relevant Source's or Destination's config modal.
5. Select **TLS Settings (Client Side)** or **TLS Settings (Server Side)**, depending on the integration.
6. Toggle **Enabled** on.
7. In the **Certificate name** drop-down, select a certificate that you've preconfigured for this integration. (This will auto-populate the corresponding fields here.)

   Or, if you're creating a new certificate, instead click the adjacent **Create** button.

1. Click **Save**.

Cribl Edge will create new certificates at the same Fleet level that you're configuring. You can verify this at the bottom of the **Create new certificate** modal, by making sure that the **Referenced** table includes

rows for poulated for the appropriate **Sources** and **Destinations**.



Group-level certificate modal

Certificate revocation can lead to unexpected service interruptions. To mitigate risks:

- To minimize impact, 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 Fleet 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, clicking **Add Header** as needed.

9.3.8. Configure System Proxy

You can direct all outbound HTTP/S requests to go through proxy servers. You do so by setting a few environment variables before starting Cribl Edge, as follows:

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.

`HTTP_PROXY` examples:

```
$ export HTTP_PROXY=http://10.15.20.25:1234
$ export HTTP_PROXY=http://proxy.example.com:1234
```

`HTTPS_PROXY` examples:

```
$ export HTTPS_PROXY=http://10.15.20.25:5678
$ export HTTPS_PROXY=http://proxy.example.com:5678
```

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.

Proxy Configuration with systemd

If you are proxying outbound traffic and starting Cribl Edge using systemd, add your proxy environment variables to the systemd override file (see Persisting Overrides). Add statements of this form:

Installed systemd File

```
[Service]
Environment=http_proxy=<yourproxy>
Environment=https_proxy=<yourproxy>
Environment=no_proxy=<no_proxy_list>
```

This will prevent Cribl Edge from throwing "failed to send anonymized telemetry metadata" errors.

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:

```
$ export HTTP_PROXY=http://username:password@proxy.example.com:1234
$ export HTTPS_PROXY=http://username:password@proxy.example.com:5678
```

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:

```
$ export 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 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.

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.

9.3.9. 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. This document will guide you through the necessary steps to set up a SOCKS proxy for your Cribl Edge deployment.

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.

`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>`. Specify the SOCKS proxy host, port, and optional authentication credentials in the variable.

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`.

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.

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**
    - **Proxy host**: Enter the proxy server host. Can be a `hostname`, `IPv4`, or `IPv6`.
    - **Proxy port**
    - **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.

9.4. Manage Secrets And Keys

9.4.1. 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.

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** > `<group-name>` **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.

9.4.2. 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.

To ensure they can be decoded correctly, all secrets in Cribl Edge must use `latin1` encryption.

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 **Groups** > `<group-name>` **Settings** > **Security** > **Secrets**.

On the **Manage Secrets** page, you can configure existing secrets, and/or click **New Secret** to define new secrets.

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 Stream 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.

9.4.3. Configure KMS

Cribl Edge's Key Management Service (KMS) maintains the keys that Cribl Edge uses to encrypt secrets on Fleets and Edge Nodes. The internal KMS is always available, but integrating an external KMS provider is available on certain plan/license tiers. For details, see Pricing..

To configure KMS for Cribl Edge in Cribl.Cloud, see the Launch Guide.

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.

The resulting **KMS Provider** 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 Edge Settings' parallel Secrets section.
- HashiCorp Vault
- AWS KMS

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. Setting an invalid region results in a fallback to the global STS endpoint.

Enable for Vault Auth: Toggle to `Yes` 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 `Yes`.

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. Setting an invalid region results in a fallback to the global STS endpoint.

Enable for KMS: Toggle to `Yes` 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.

10. 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.

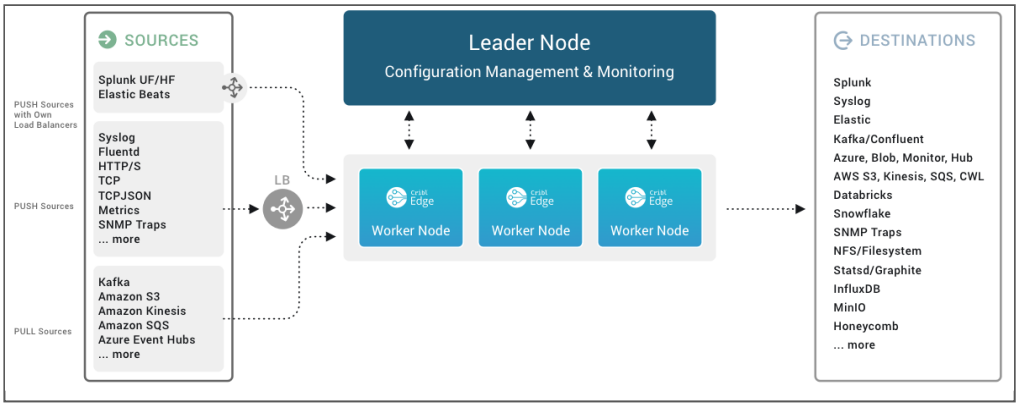
Key Concepts for Optimal Data Flow

- 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.

10.1. Sources

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.



Sources in the Edge ecosystem

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.

- AppScope
- 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 Cribl Edge **fetches** data from.

- Amazon Data Firehose
- Datadog Agent
- Elasticsearch API
- Grafana
- HTTP/S (Bulk API)
- Loki
- Metrics
- Model Driven Telemetry
- Netflow
- Raw HTTP/S
- OpenTelemetry (OTel)
- Prometheus Remote Write
- SNMP Trap
- Splunk HEC
- Splunk TCP
- Syslog
- TCP JSON
- TCP (Raw)
- UDP (Raw)
- Windows Event Forwarder

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.

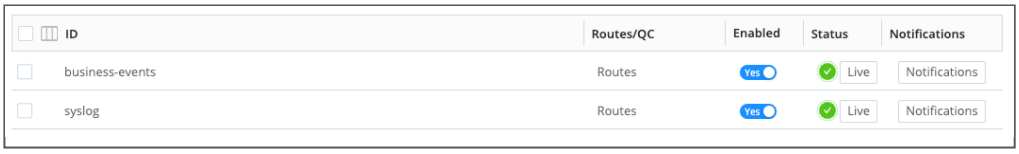
To configure Sources, from the top nav, click **Manage**, then select a **Fleet** to configure. Then, you have two options:

- To access the graphical QuickConnect UI, click **Collect**. Next, click either **Add New** or (if displayed) Select **Existing**.
- To access the Routing UI, click **More** > **Sources**. On the resulting **Data Sources** page's tiles or left menu, select the desired type, then click **Add New**.

Capturing Source Data

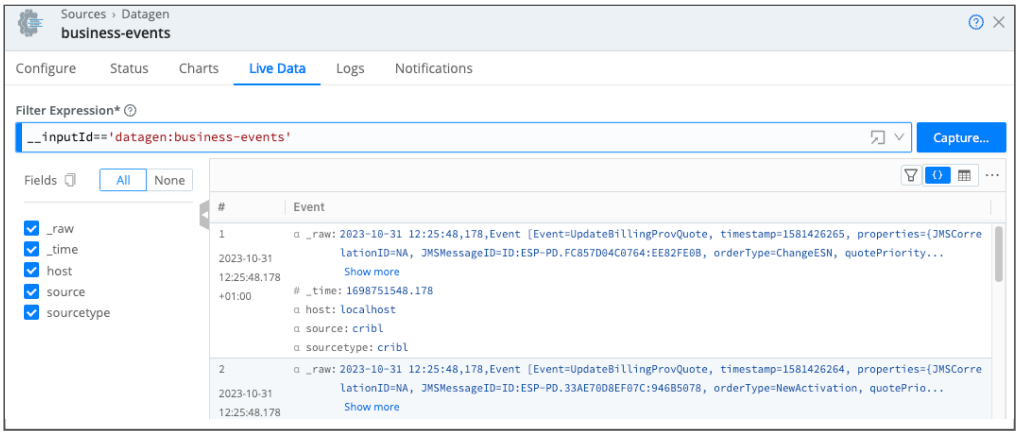
To capture data from a single enabled Source, you can bypass the Preview pane, and instead capture directly from a **Manage Sources** page. Just click the **Live** button beside the Source you want to capture.

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 clicking the modal's **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.
	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.

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 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 visualization of the recent activity on the Source. The following data is available:

- Events in
- Thruput in (events per second)
- Bytes in
- Thruput in (bytes per second)

This data (in contrast with the status tab) is read almost instantly and does not reset when restarting an Edge Node.

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.

- **System Metrics** – Basic Level
- **File Monitor** > **in_file_auto** – Auto Discovery Mode
- **File Monitor** > **in_file_varlog** – Manual Discovery Mode
- **AppScope** > **in_appscope** – Unix Domain Socket listener
- **Cribl Internal** > **CriblLogs** – Internal
- **Cribl Internal** > **CriblMetrics** – Internal

10.1.1. System

10.1.1.1. 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.

Available in Cribl.Cloud: **No** | Type: **System** | TLS Support: **N/A** | Event Breaker Support: **Yes**

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 `Yes`.
    - **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**:
    - **Max retries**: 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), maxiumum `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 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 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.

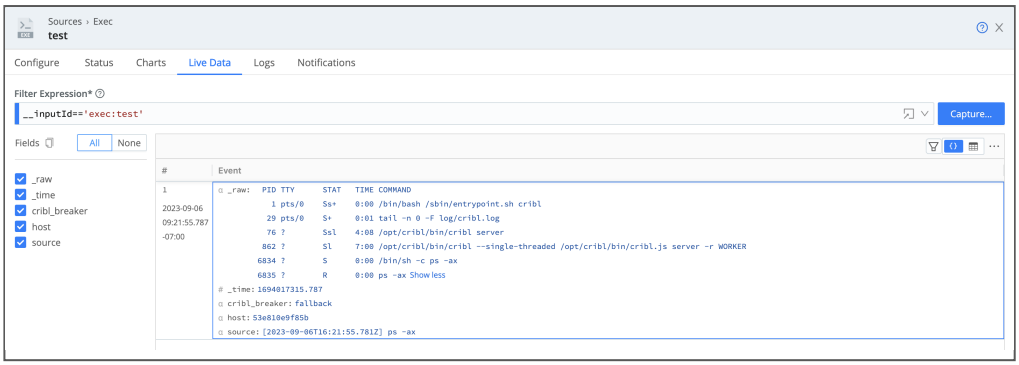
Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

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

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.

10.1.1.2. 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.

Available in Cribl.Cloud: **No** | Type: **System** | TLS Support: **N/A** | Event Breaker Support: **Yes**

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.
- **ASCII**: `text`.
- **Non-ASCII binary**.

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. It can be configured to 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 an **Allowed list** 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 path/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.

- 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*`.

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 to `Yes` 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**: Use the buttons to select one of these options: **Auto** or **Manual**. For details, see Discovery Mode below.

3. Next, you can configure the following **Optional Settings**:
   - **Filename allowlist**: Wildcard syntax, and the exclamation mark (`!`) for negation, are allowed. For example, you can use `!*cribl*access.log` to prevent the Source from discovering Cribl Edge's own log files. The default filters are `*/log/*` and `*log`.
   - **Age duration limit**: 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**: If toggled to `Yes`, this Source will skip files with modification times older than the configured **Age duration limit**. When you set an **Age duration limit** threshold, the Source will open every file and read through all its contents to seek a timestamp. For best performance, also enable the **Check file modification times** toggle to skip older files.
   - **Collect from end**: If toggled to `Yes`, this Source will prioritize processing the latest data by jumping to the end of newly discovered files on Cribl Edge's startup. This is ideal if you want to focus 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 its normal processing mode, reading from the beginning to capture ongoing updates.
   - **Enable binary files**: If toggled to `Yes`, the Source will 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**: If toggled to `Yes`, the Source displays 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**.

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 below.

Discovery Mode

The File Monitor Source offers two options:

Auto: Tells the Source to automatically discover files that running processes have open for writing.

Manual: Tells the Source to discover the files within the **Search path** (a directory) that you specify, down to the **Max depth**. You must enter an absolute path (no wildcards) for the **Search path**, as File Monitor searches files in a single directory in Manual mode. The defaults are:

- **Search path** is set to the `/var/log` directory
- **Allowlist** of `*/log/*` and `*log`
- **Max depth** of 4

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.

Both modes allow you to set the **Polling interval**, which otherwise defaults to 10 seconds.

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.), maxiumum `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 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 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.

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.

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`

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.

Configuring 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.

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 license issues.

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

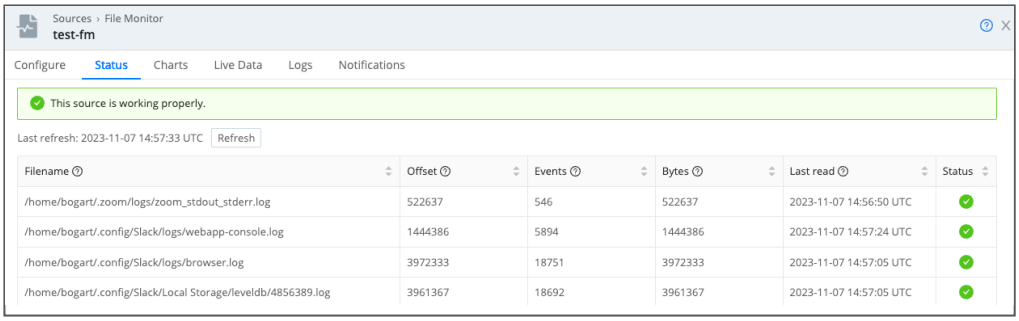
Recall that you'll often create a set of File Monitor Sources, each of which monitors one file. This is **not** the right approach in an example like this one, where **Discovery Mode** is set to **Auto**. Using **Auto** mode in a file monitor along with literally any other file monitor can cause unintended, 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`	`*`

10.1.1.3. 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"
```

Also, this Source does not support the new `COMPACT` binary file format. The logs will contain an error message when the parser encounters this type of file.

Configure a Journal Files Source

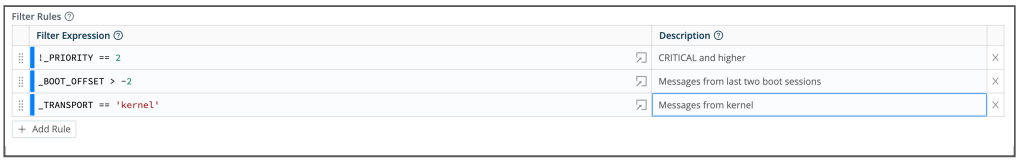
1. On the top bar, select **Products**, and then select **Cribl Edge**. Under **Fleets**, select a Fleet. Next, you have two options:
    - To configure via QuickConnect, navigate to **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Select **Add Source** and select the Source you want from 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 to `Yes` 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 add Fields to each event, using Eval-like functionality.

Name: Field name.

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 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.

10.1.1.4. 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, or 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**

Cribl Edge supports configuring only one Kubernetes Events Source per Edge Node and per Fleet.

On Cribl.Cloud, this Source is available on customer-managed hybrid Workers, but Cribl-managed Workers in Cribl.Cloud omit this Source – Cribl manages these Workers' uptime and diagnostics on your behalf.

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, 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**: 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.
3. 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.
4. Optionally, configure any Processing and Advanced settings, or Connected Destinations outlined in the sections below.
5. 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. The default filter is: `!metadata.namespace.startsWith('kube-')`, which ignores Pods in the `kube-*` namespace.

Other **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 add Fields to each event using Eval-like functionality.

Name: Field name.

Value: JavaScript expression to compute field's value, enclosed in quotes or backticks. (Can evaluate to a constant.)

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

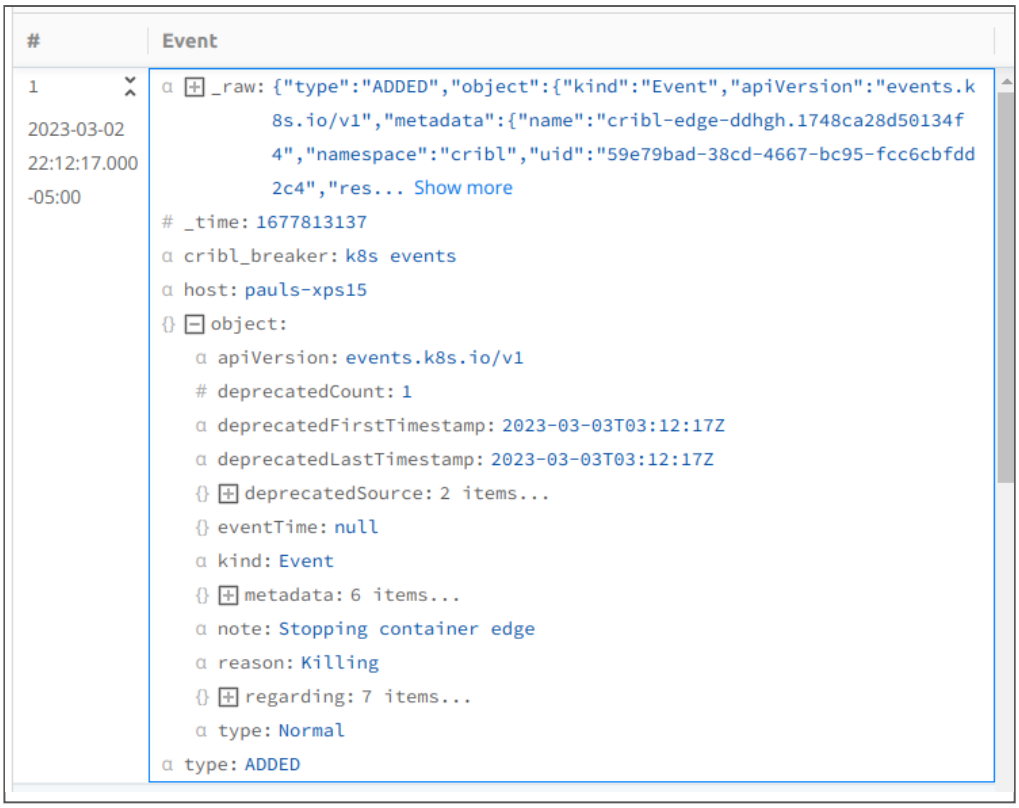
Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

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.

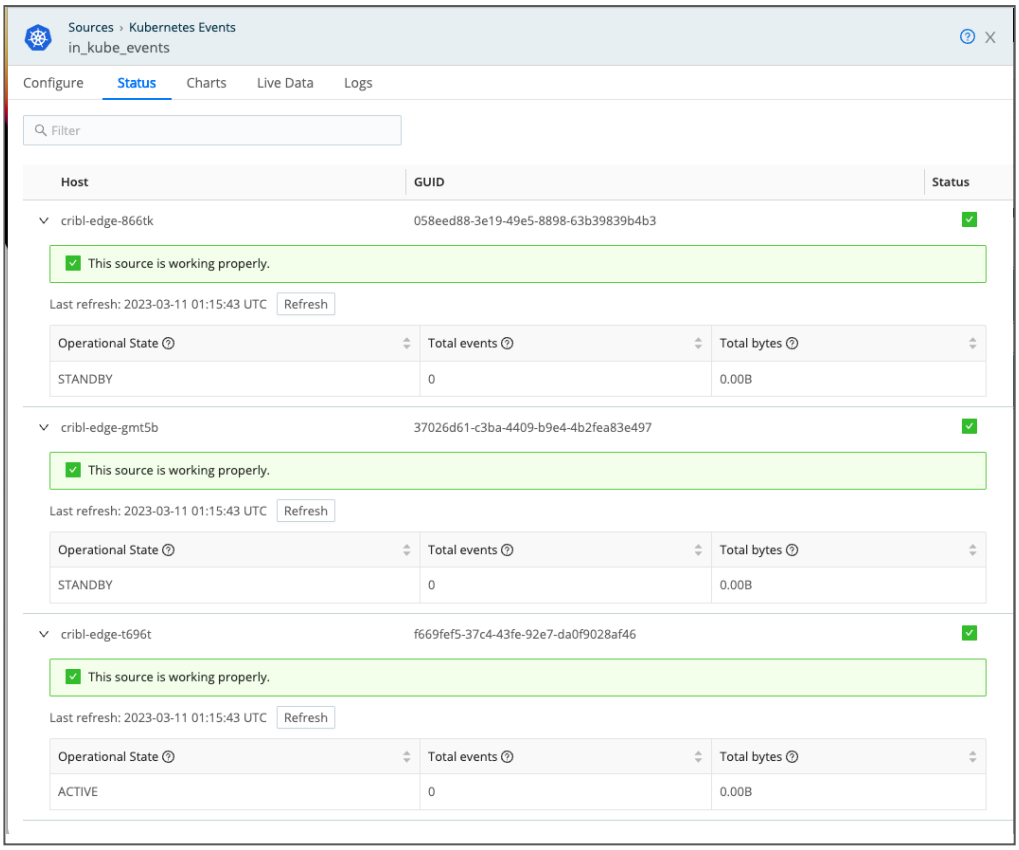
If you are sending the events over a metered connection, then you will get three copies of the same data: `_raw`, `__raw`, and the extracted `object` fields. Consider dropping the redundant fields before sending them to your Destination(s).



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.

Sample Event

10.1.1.5. 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

Cribl Edge supports configuring only one Kubernetes Logs Source per Edge Node and per Fleet. This Source is currently unavailable on Cribl-managed Workers in Cribl.Cloud.

Discover and Filter Kubernetes Pods to Monitor

The Kubernetes Logs Source connects to the Kubernetes API and loads the lists of Pods on the node, 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.

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, 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**: 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.
3. Next, you can configure the following **Optional Settings**:
   - **Polling interval**: How often, in seconds, to collect metrics. If not specified, defaults to `15` seconds.
   - **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 to `Yes`, 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.
4. Optionally, configure any Processing, Disk Spooling, and Advanced settings, or Connected Destinations outlined in the sections below.
5. 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.

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 add Fields to each event using Eval-like functionality.

Name: Field name.

Value: JavaScript expression to compute field's value, enclosed in quotes or backticks. (Can evaluate to a constant.)

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.

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.

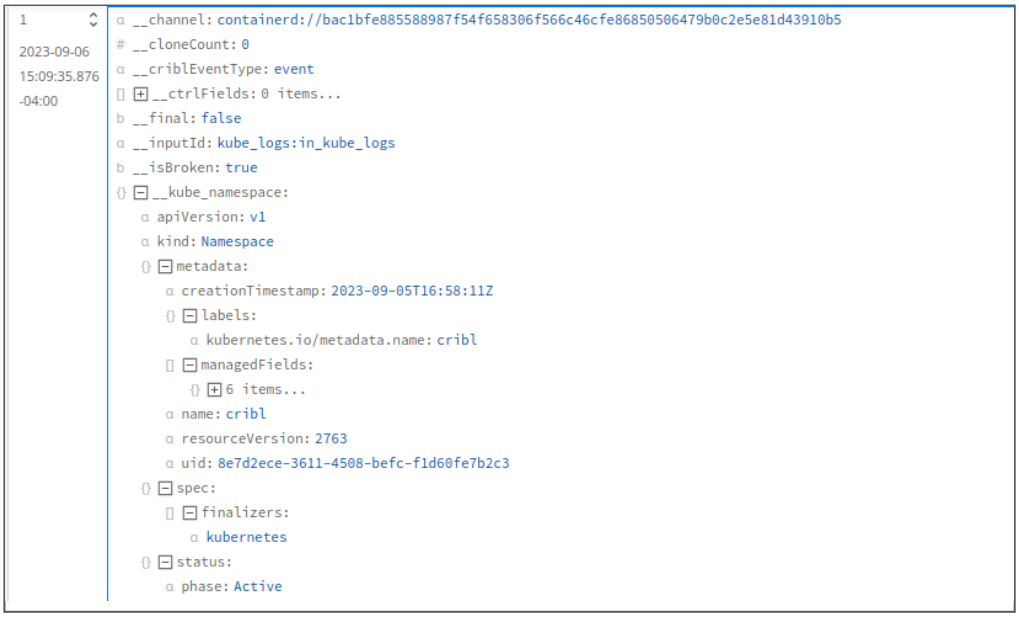
Key-Value Store

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

10.1.1.6. 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 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**

Cribl Edge supports configuring only one Kubernetes Metrics Source per Edge Node and per Fleet. This Source is currently unavailable on Cribl-managed Workers in Cribl.Cloud.

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, 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**: 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.
3. 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.
4. Optionally, configure any Processing, Disk Spooling, and Advanced settings, or Connected Destinations outlined in the sections below.
5. 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 add Fields to each event using Eval-like functionality.

Name: Field name.

Value: JavaScript expression to compute field's value, enclosed in quotes or backticks. (Can evaluate to a constant.)

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 to `Yes` 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.

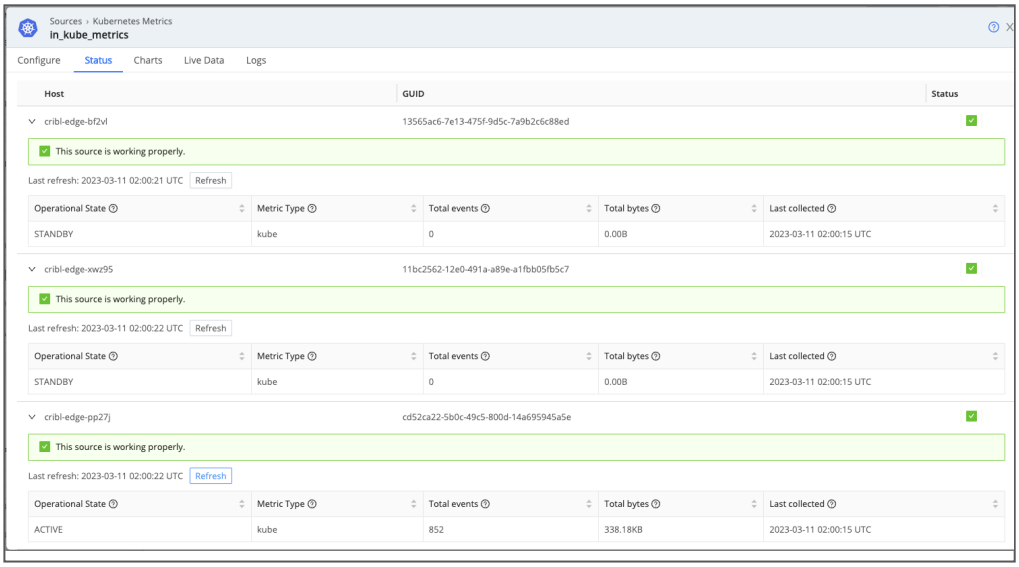
Connected Destinations

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

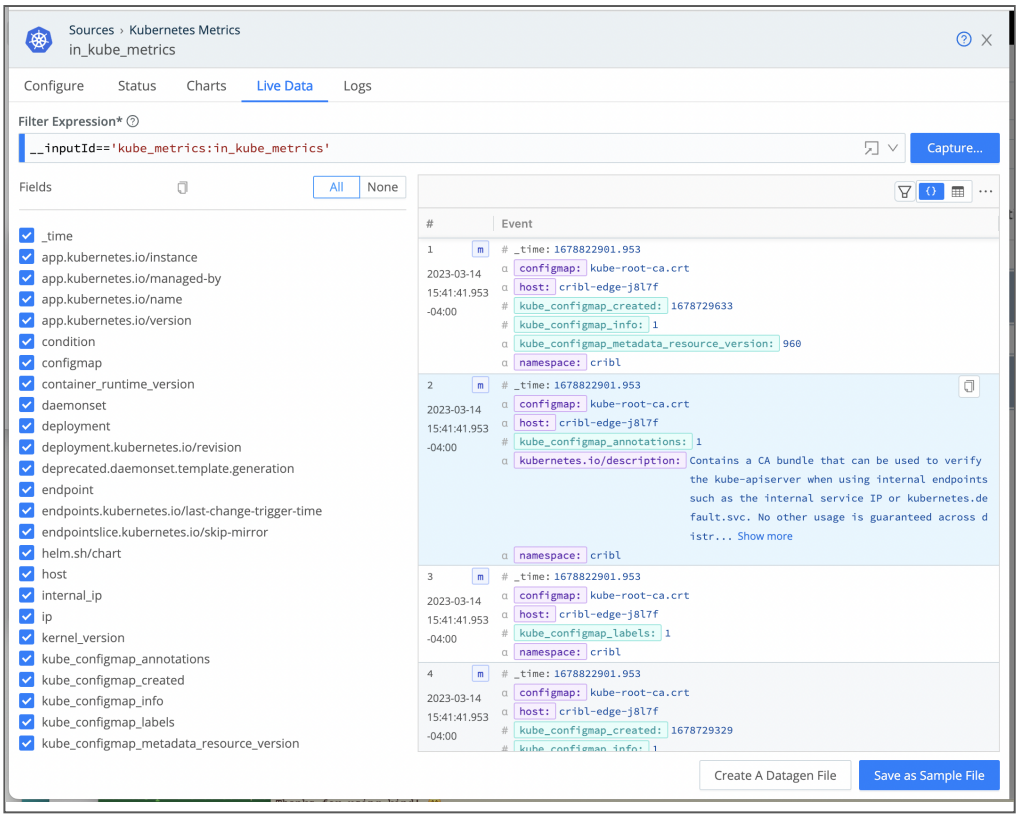
To check the Source's operational state, go to **Status** and expand the host details. The **Operational State** Column shows either an **Active** state or **Standby**. **Active** indicates the Source is running or won the election. **Standby** means it's waiting to be re-elected and not currently running.



Operational State

Source Live Data

To see a sample of the Source's data, select the **Live Data** tab.


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
kube_configmap_info
kube_configmap_labels
kube_configmap_metadata_resource_version
#
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
2023-03-14 15:41:41.953 -04:00
_time: 1678822901.953
configmap: kube-root-ca.crt
host: cribl-edge-j8l7f
kube_configmap_created: 1678729329
Create A Datagen File
Save as Sample File

Live Data

10.1.1.7. 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 not available 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 the Routes, select **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). Select the Source you want. Next, select **Add Source**.
2. Next, select the default `in_system_metrics` to open the configuration modal.
3. Configure the following under **General Settings**:
    - **Enabled**: Toggle to `Yes` to enable the Source.
    - **Input ID**: This is prefilled with the default value `in_system_metrics`, which cannot be changed via the UI.
    - **Description**: Optionally, enter a description.
4. 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.
5. Optionally, you can configure metrics for Host, Process, Container, or adjust Processing, Advanced settings and Connected Destinations outlined in the sections below.
6. 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. Click 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 children**: When toggled to “Yes”, 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, click **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 Filters 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.

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.

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 to `Yes` to include stopped and paused containers.
- **Per device metrics**: Toggle to `Yes` to generate separate metrics for each device.
- **Detailed metrics**: Toggle to `Yes` 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 add Fields to each event using Eval-like functionality.

Name: Field name.

Value: JavaScript expression to compute field's value, enclosed in quotes or backticks. (Can evaluate to a constant.)

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 persistence: Whether to save metrics to disk. When set to `Yes`, 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/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.

10.1.1.8. 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**

Cribl Edge supports configuring only one System State Source per Edge Node and per Fleet.

This Source is currently unavailable on Cribl-managed Workers in Cribl.Cloud.

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 the Routes, select **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). Select the Source you want. Next, select **Add Source**.
2. Select the default `in_system_state` to open the configuration modal.
3. Configure the following under **General Settings**:
    - **Enabled**: Toggle to `Yes` to enable the Source.
    - **Input ID**: This is prefilled with the default value `in_system_state`, 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 to collect metrics, in seconds. Defaults to `10`. 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.
5. Optionally, you can adjust the Collector, Processing, and Advanced settings, or Connected Destinations outlined in the sections below.
6. 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`
- `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.

This option is available for Windows and Linux Fleets, but you must create a different Fleet for each OS. Follow this procedure:

1. Create a Fleet for the appropriate OS.
2. Modify the Fleet by disabling or removing the pre-defined Sources that do not apply to the OS. Remove Windows Event Forwarder or Windows Event Logs and Windows Metrics from Linux Fleets, and remove Linux-specific Sources from Windows Fleets.
3. Modify the Fleet mapping to filter by platform – `platform=='win32'` for Windows or `platform=='linux'` for Linux.
4. Deploy Cribl Edge Edge Nodes and verify that the mapping is correct.

If you are using the free version of Cribl Edge, you will only get a single default Fleet. If you need more than one Fleet, you'll have to limit your Cribl Edge deployment to one OS or upgrade to Cribl Edge Enterprise.

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 (mountpoint)

For each mounted filesystem, Cribl Edge will collect:

- mountpoint (path)
- disk (name)
- partition (ID)
- filesystem type
- size (in blocks)
- blockSize (in bytes)
- blocksFree
- inodes
- inodesFree
- flags

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.

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 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.

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

On 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: 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.

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 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.

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.

Users and Groups

With the **Enabled** toggle on (the default), Cribl Edge will create events for users and groups.

Linux

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

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 add Fields to each event using Eval-like functionality.

Name: Field name.

Value: JavaScript expression to compute field's value, enclosed in quotes or backticks. (Can evaluate to a constant.)

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 set to `Yes`, 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

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.

10.1.2. Internal

10.1.2.1. Cribl HTTP Source

The Cribl HTTP Source receives data from a Cribl HTTP Destination connected to the same Leader. Relaying data between a Cribl HTTP Destination and Cribl HTTP Source pair prevents double-billing: You'll be charged no additional charges or credits after initial ingest to the first Fleet.

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.

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. All data subsequently relayed to other Workers via a Cribl HTTP Destination/Source pair is not charged.

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 Cribl HTTP Destination must be on a Node that is connected to the same Leader as the Cribl HTTP Source(s).
- 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 later can send data only to Nodes running v.3.5.4 and later. Nodes running the Cribl HTTP Source on Cribl 3.5.3 and earlier can send data only to Nodes running v.3.5.3 and earlier.

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 to `Yes` 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)

Enabled Defaults to `No`. When toggled to `Yes`, 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. Defaults to `No`. When toggled to `Yes`, exposes these two additional fields:

- **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). Defaults to `No`.
- **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 `No`. When toggled to `Yes`:

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.

Max buffer size: 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.

Max file size: 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`.

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, 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 add Fields to each event, using Eval-like functionality.

Name: Field name.

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 to process data from this input before the data is sent through the Routes.

Advanced Settings

Show originating IP: Toggle to `Yes` 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 this to `Yes` to add request headers to events, in the `__headers` field.

Health check endpoint: Toggle to `Yes` 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.

Max active requests: 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.

Max requests per socket: 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

Max requests per socket 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 **Max requests per socket** 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 **Max requests per socket**, 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 **Max requests per socket**.

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 set to `Yes`.
`__inputId`
`__outputId`
`__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 set to `No`, the original client IP/port in this header will override the value of `__srcIpPort`.

If **Show originating IP** is set to `Yes`, 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.

10.1.2.2. Cribl TCP Source

The Cribl TCP Source receives data from a Cribl TCP Destination connected to the same Leader. Relaying data between a Cribl TCP Destination and Cribl TCP Source pair prevents double-billing: You'll be charged no additional charges or credits after initial ingest to the first Fleet.

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. All data subsequently transferred to other Workers via a Cribl TCP Destination/Source pair is not charged.

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:

- The Cribl TCP Destination must be on a Node that is connected to the same Leader as the Cribl TCP Source.
- 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 later can send data only to Nodes running v.3.5.4 and later. Nodes running the Cribl TCP Source on Cribl 3.5.3 and earlier can send data only to Nodes running v.3.5.3 and earlier.

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 to `Yes` 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)

Enabled defaults to `No`. When toggled to `Yes`:

Certificate name: 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. Defaults to `No`. When toggled to `Yes`:

- **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). Defaults to `No`.
- **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 `No`. When toggled to `Yes`:

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.

Max buffer size: 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`.

Max file size: 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 add Fields to each event, using Eval-like functionality.

Name: Field name.

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 to process data from this input before the data is sent through the Routes.

Advanced Settings

Enable proxy protocol: Toggle to `Yes` if the connection is proxied by a device that supports proxy protocol v1 or v2.

Enable load balancing (Cribl Stream only): Toggle to `Yes` 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 only in Cribl Stream Distributed deployments.

Max active connections: 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.

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`

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.

10.1.2.3. Cribl Stream Source (Deprecated)

This Source was deprecated as of Cribl Edge 3.5, and has been removed as of v.4.0. Please instead use the Cribl TCP or the Cribl HTTP Source to enable Edge Nodes to send data to peer Nodes.

The Cribl Stream internal Source is available only in distributed deployments. It is provided to facilitate sending data between Edge Nodes that are connected to the same Leader. You'll find this Source especially valuable in a hybrid Cloud deployment.

This Source can receive data only from a TCP JSON Destination, on a Edge Node that is connected to the same Leader. The following instructions cover configuring such a TCP JSON Destination.

Type: **Internal** | TLS Support: **YES** | Event Breaker Support: **No**

Use New Sources Instead

The replacement Sources, Cribl TCP and the Cribl HTTP, don't rely on IP filtering, for the following reasons:

- Load balancers and/or proxies between the Cribl Destination and Cribl Source can change the IP address, resulting in a bad match and rejected ingest.
- A Lookup table of all IP addresses needed to be sent to each Worker Node/Edge Node from the Leader, which is not scalable.
- The Lookup table of IP addresses required constant communication between the Worker Node/Edge Nodes and the Leader, making this fragile and placing an arbitrary reliance on the Leader that shouldn't be there.

How It Works

You can use the Cribl Stream Source to send data between Workers. In a hybrid Cribl.Cloud deployment, using this Source ensures that you're billed for ingress only once – when the data is originally received. All other data transferred into Workers via the Cribl Stream Source is not charged.

This use case is common in deployments where an on-prem Worker sends data to a Worker in Cribl Cloud, for additional processing and then routing to Destinations.

As one usage example, assume that you want to send data from one Worker Node/Edge 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 TCP JSON Destination.
- Create a "Cribl Stream" Source, on the target Worker Group/Fleet in Cribl Cloud.
- Configure these to send data from the File System Collector to the TCP JSON Destination, and from there to the "Cribl Stream" Source in Cribl Cloud.

The key points about configuring this architecture are:

- The TCP JSON Destination must be on a Worker Node/Edge Node that is connected to the same Leader as the "Cribl Stream" Source.
- The TCP JSON Destination's **Address** and **Port** must match the "Cribl Stream" Source's **Address** and **Port**.

Configuring the Cribl Stream Source

In the **QuickConnect** UI: Click **New Source** or **Add Source**. From the resulting drawer's tiles, select [**System and Internal** >] **Cribl Stream**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The drawer will now provide the following options and fields.

Or, in the **Data Routes** UI: From the top nav of a Cribl Stream instance or Group, select **Data** > **Sources**. From the resulting page's tiles or the **Sources** left nav, select [**System and Internal** >] **Cribl Stream**. Next, click **New Source** to open a **Cribl Stream** > **New Source** modal that provides the following options and fields.

General Settings

Input ID: Enter a unique name to identify this Cribl Stream Source definition. If you clone this Source, Cribl Edge will add `-CLONE` to the original **Input ID**.

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.

Authentication Settings

Use the **Authentication method** buttons to select one of these options:

- **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, unauthenticated 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 Stream's internal secrets manager or (if enabled) in an external KMS. A **Create** link is available if you need a new secret.

Optional Settings

Tags: Optionally, add tags that you can use for filtering and grouping in the Cribl Stream UI. Use a tab or hard return between (arbitrary) tag names. These tags aren't added to processed events.

TLS Settings (Server Side)

Enabled defaults to `No`. When toggled to `Yes`:

Certificate name: 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. Defaults to `No`. When toggled to `Yes`:

- **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). Defaults to `No`.
- **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 this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

> On Cribl-managed Cloud Workers (with certain plan/license tiers), 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 no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

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 Cribl Edge's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Edge's data processing engine.

Max buffer size: 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 per connection, not just per Worker Process – and this can dramatically expand memory usage.)

Commit frequency: The number of events to send downstream before committing that Stream has read them. Defaults to `42`.

Max file size: 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, etc. If not specified, Cribl Edge applies the default `1 MB`.

Max queue size: 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`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in 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 Edge will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

In Cribl Edge 4.1 and later, Source-side PQ's default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Always On versus Smart Mode.

You can optimize Workers' startup connections and CPU load at **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 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 to process data from this input before the data is sent through the Routes.

Advanced Settings

Enable Proxy Protocol: Toggle to `Yes` 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. Although it appears in the **Advanced Settings** UI, this is not a user-configurable setting. Its value is updated automatically to match the IP addresses used in the deployment, and cannot be edited.

Max Active Connections: Maximum number of active connections allowed per Worker Process. Use `0` for unlimited.

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 Stream 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`

10.1.2.4. 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**

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, choosing **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. 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. Keep it at `Yes` to enable Cribl logs as a Source.
        - **Input ID**: This is prefilled with the default value `CriblLogs`, which cannot be changed via the UI.
        - **Description**: Optionally, enter a description.
    - For Cribl Metrics, configure the following:
        - **Enabled**: This duplicates the parent page's **Enabled** toggle. Keep it at `Yes` to enable Cribl metrics as a Source.
        - **Input ID**: This is prefilled with the default value `CriblLogs`, which cannot be changed via the UI.
        - **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 this to `No` to exclude granular metrics that can cause high CPU load. For details, see Full Fidelity below. .
    - **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.

If Cribl Edge detects `source` or `host` fields in metrics, it copies their values into new dimensions with added `event_` prefixes (for example, `event_source`). This preserves the original fields' values if they're overwritten in downstream services. For details, see Duplicated Fields/Dimensions.

You can disable metric collection for these and other fields by specifying them in **Settings** > **Global** > **System** > **General Settings** > **Limits** > **Metrics** > **Disable field metrics**.

5. Optionally, configure any Processing and Advanced settings outlined in the sections below.
6. Select **Save**, then **Commit & Deploy**.

Full Fidelity

When you toggle **Full fidelity** to `No`, the following metrics will be dropped:

- `cribl.logstream.host.(in_bytes,in_events,out_bytes,out_events)`
- `cribl.logstream.index.(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)`

Processing Settings

Fields

In this section, you can add Fields to each event, using Eval-like functionality. For example, here you could specify adding an `index` field.

Name: Field name.

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 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.

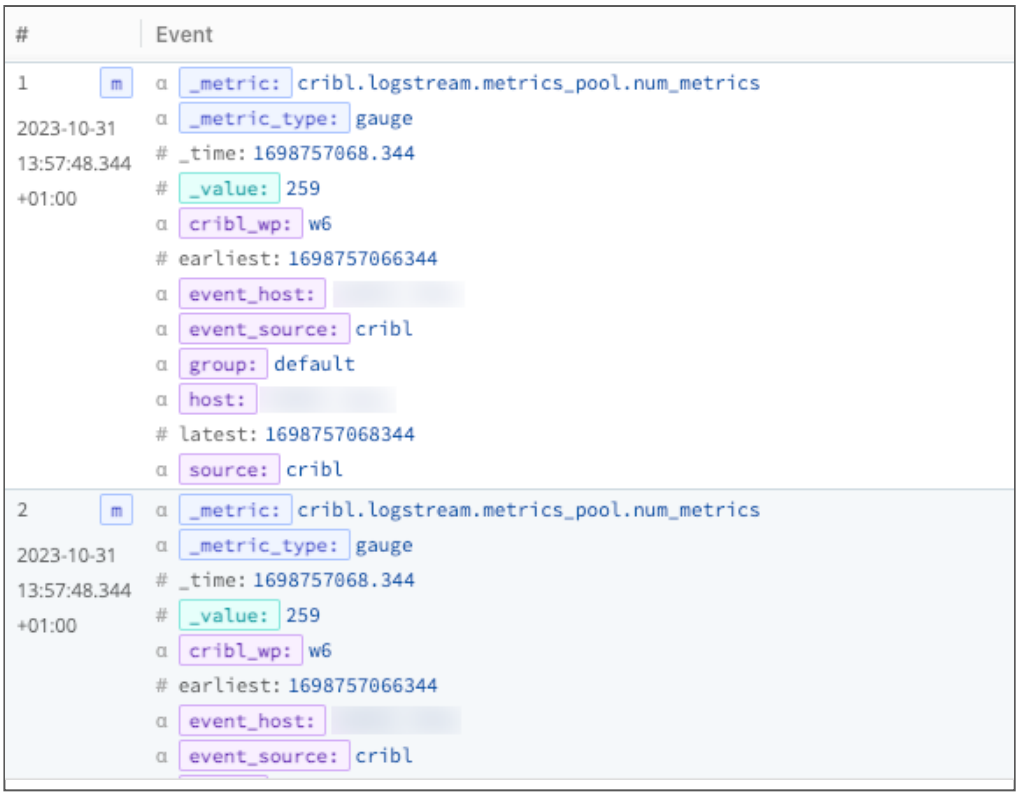
To do this: In the same `cribl_metrics_rollup` pre-processing Pipeline (or a clone) that you attach to your Source, enable the final Eval Function, which applies this **Filter** expression to remove the `sourcetype` field: `_metric && _metric.startsWith('cribl.logstream.sourcetype.')`

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.

10.1.2.5. 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 add Fields to each event using Eval-like functionality.

Name: Field name.

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 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`

10.1.3. Amazon

10.1.3.1. 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.<your-org-id>.cribl.cloud` on port `443`. Replace `<your-org-id>` 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 to `Yes` 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**:
   - **Auth tokens**: Shared secrets to be provided by any client (Authorization: <token>). Click **Generate** to create a new secret. If empty, unauthenticated 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 `No`. When toggled to `Yes`:

Certificate name: 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. Defaults to `No`. When toggled to `Yes`:

- **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). Defaults to `No`.
- **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 this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

> On Cribl-managed Cloud Workers (with certain plan/license tiers), 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 no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

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 Cribl Edge's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Edge's data processing engine.

Max buffer size: 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 per connection, not just per Worker Process – and this can dramatically expand memory usage.)

Commit frequency: The number of events to send downstream before committing that Stream has read them. Defaults to `42`.

Max file size: 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, etc. If not specified, Cribl Edge applies the default `1 MB`.

Max queue size: 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`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in 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 Edge will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

In Cribl Edge 4.1 and later, Source-side PQ's default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Always On versus Smart Mode.

You can optimize Workers' startup connections and CPU load at **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 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 to process data from this input before the data is sent through the Routes.

Advanced Settings

Show originating IP: Toggle to `Yes` 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 this to `Yes` to add request headers to events, in the `__headers` field.

Health check endpoint: Toggle to `Yes` 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.

Max active requests: 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.

Max requests per socket: 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

Max requests per socket 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 **Max requests per socket** 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 **Max requests per socket**, 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 **Max requests per socket**.

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 set to `Yes`.
- `__inputId`
- `__raw`
- `__srcIpPort` – See details below.

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 set to `No`, the original client IP/port in this header will override the value of `__srcIpPort`.

If **Show originating IP** is set to `Yes`, 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.

10.1.4. Prometheus

10.1.4.1. 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.

In addition to the functionality already supported in the existing Prometheus Scraper, this Source is designed to work seamlessly in Kubernetes environments; and no longer uses internal jobs framework allowing it to handle large-scale Cribl Edge deployments.

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**: The `Yes` default uses the public IP address for discovered targets. Toggle to `No` 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 to `Yes` 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. **IAM Roles for Amazon EC2**: Loaded from AWS Identity and Access Management (IAM) roles attached to an EC2 instance.
2. **Shared Credentials File**: Loaded from the shared credentials file (`~/.aws/credentials`).
3. **Environment Variables**: Loaded from environment variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`.
4. **JSON File on Disk**: Loaded from a JSON file on disk.
5. **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 add Fields to each event using Eval-like functionality.

Name: Field name.

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 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 set to `Yes`, 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. The default setting (`Yes`) 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 `Yes`, 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.

10.1.4.2. 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.

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 to `Yes` 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 `No`. When toggled to `Yes`:

Certificate name: 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. Defaults to `No`. When toggled to `Yes`:

- **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). Defaults to `No`.
- **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 this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

> On Cribl-managed Cloud Workers (with certain plan/license tiers), 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 no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

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 Cribl Edge's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Edge's data processing engine.

Max buffer size: 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 per connection, not just per Worker Process – and this can dramatically expand memory usage.)

Commit frequency: The number of events to send downstream before committing that Stream has read them. Defaults to `42`.

Max file size: 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, etc. If not specified, Cribl Edge applies the default `1 MB`.

Max queue size: 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`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in 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 Edge will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

In Cribl Edge 4.1 and later, Source-side PQ's default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Always On versus Smart Mode.

You can optimize Workers' startup connections and CPU load at **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 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 to process data from this input before the data is sent through the Routes.

Advanced Settings

Show originating IP: Toggle to `Yes` 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 this to `Yes` to add request headers to events, in the `__headers` field.

Health check endpoint: Toggle to `Yes` 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.

Max active requests: 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.

Max requests per socket: 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

Max requests per socket 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 **Max requests per socket** 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 **Max requests per socket**, 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 **Max requests per socket**.

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 set to `Yes`.
- `__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 set to `No`, the original client IP/port in this header will override the value of `__srcIpPort`.

If **Show originating IP** is set to `Yes`, 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.

10.1.4.3. 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 to `Yes` 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**:
    - **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`.
    - **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 `No`. When toggled to `Yes`:

Certificate name: 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. Defaults to `No`. When toggled to `Yes`:

- **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). Defaults to `No`.

- **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 this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

> On Cribl-managed Cloud Workers (with certain plan/license tiers), 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 no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

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 Cribl Edge's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Edge's data processing engine.

Max buffer size: 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 per connection, not just per Worker Process – and this can dramatically expand memory usage.)

Commit frequency: The number of events to send downstream before committing that Stream has read them. Defaults to `42`.

Max file size: 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, etc. If not specified, Cribl Edge applies the default `1 MB`.

Max queue size: 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`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in 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 Edge will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

In Cribl Edge 4.1 and later, Source-side PQ's default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Always On versus Smart Mode.

You can optimize Workers' startup connections and CPU load at **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 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 to process data from this input before the data is sent through the Routes.

Advanced Settings

Show originating IP: Toggle to `Yes` 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 this to `Yes` to add request headers to events, in the `__headers` field.

Health check endpoint: Toggle to `Yes` 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.

Max active requests: 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; and so forth.

Max requests per socket: 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

Max requests per socket 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 **Max requests per socket** 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 **Max requests per socket**, 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 **Max requests per socket**.

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 set to `Yes`.
- `__inputId`
- `__labels` – For log events only – will contain all the labels found in each event's corresponding Loki stream.
- `__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 set to `No`, the original client IP/port in this header will override the value of `__srcIpPort`.

If **Show originating IP** is set to `Yes`, 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.

10.1.4.4. 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 to `Yes` 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`.
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 `No`. When toggled to `Yes`:

Certificate name: 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. Defaults to `No`. When toggled to `Yes`:

- **Validate server certs**: Toggle to `Yes` 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 this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

> On Cribl-managed Cloud Workers (with certain plan/license tiers), 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 no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

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 Cribl Edge's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Edge's data processing engine.

Max buffer size: 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 per connection, not just per Worker Process – and this can dramatically expand memory usage.)

Commit frequency: The number of events to send downstream before committing that Stream has read them. Defaults to `42`.

Max file size: 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, etc. If not specified, Cribl Edge applies the default `1 MB`.

Max queue size: 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`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in 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 Edge will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

In Cribl Edge 4.1 and later, Source-side PQ's default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Always On versus Smart Mode.

You can optimize Workers' startup connections and CPU load at **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 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 to process data from this input before the data is sent through the Routes.

Advanced Settings

Show originating IP: Toggle to `Yes` 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 this to `Yes` to add request headers to events, in the `__headers` field.

Health check endpoint: Toggle to `Yes` 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.

Max active requests: 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; and so forth.

Max requests per socket: 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

Max requests per socket 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 **Max requests per socket** 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 **Max requests per socket**, 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 **Max requests per socket**.

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 set to `Yes`.
- `__inputId`
- `__labels` – Will contain all the labels found in each event's corresponding Loki stream.
- `__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 set to `No`, the original client IP/port in this header will override the value of `__srcIpPort`.

If **Show originating IP** is set to `Yes`, 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.

10.1.5. Splunk

10.1.5.1. 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.

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`, health checks via `/health`, raw events via `/raw`, and Splunk S2S events via `/s2s`. See the Splunk REST API endpoints documentation and the examples below. 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.
   - **Splunk HEC acks**: Whether to enable Splunk HEC acknowledgments. Defaults to `No`. 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** > **Splunk HEC acks**:

- When **Splunk HEC acks** is turned 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 **Splunk HEC acks** is turned 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 **Splunk HEC acks** on 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)

Enabled defaults to `No`. When toggled to `Yes`:

Certificate name: 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. Defaults to `No`. When toggled to `Yes`:

- **Validate client 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 `No`.
- **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 this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

> On Cribl-managed Cloud Workers (with certain plan/license tiers), 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 no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

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 Cribl Edge's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Edge's data processing engine.

Max buffer size: 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 per connection, not just per Worker Process – and this can dramatically expand memory usage.)

Commit frequency: The number of events to send downstream before committing that Stream has read them. Defaults to `42`.

Max file size: 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 Edge applies the default `1 MB`.

Max queue size: 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 **Max PQ size per Worker Process** in 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 Edge will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

In Cribl Edge 4.1 and later, Source-side PQ's default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Always On versus Smart Mode.

You can optimize Workers' startup connections and CPU load at **Fleet Settings** > Worker Processes.

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 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 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 **unauthenticated 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 set to `Yes`, the specified auth token is active and can be used for client access. If it's set to `No`, 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): Leave the default toggle set to `Yes` 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: When set to `Yes`, this setting drops Splunk control fields such as `crcSalt` and `savedPort`. Defaults to `Yes`. Toggle to `No` if you want to retain these fields and store the data in the `__ctrlFields` internal field.

Extract metrics (S2S only): Toggle to `Yes` 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 to `Yes` 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 later.

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 to `Yes` 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 this to `Yes` to add request headers to events, in the `__headers` field.

Max active requests: 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; and so forth.

Max requests per socket: 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

Max requests per socket 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 **Max requests per socket** 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 **Max requests per socket**, 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 **Max requests per socket**.

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 set to `Yes`.
- `__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 set to `No`, the original client IP/port in this header will override the value of `__srcIpPort`.

If **Show originating IP** is set to `Yes`, 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: myT

curl -k http://<myCriblHost>:8088/services/collector -H 'Authorization: myToken42

# Multiple Events
curl -k http://<myCriblHost>:8088/services/collector -H 'Authorization: myToken42

# Metrics Events
curl -k http://<myCriblHost>:8088/services/collector/event -H 'Authorization: myT

curl -k http://<myCriblHost>:8088/services/collector/event -H 'Authorization: myT

# Send the auth token as a query parameter, with no additional configuration
curl -k "http://<myCriblHost>:8088/services/collector/event?token=mToken42" -d '{
```

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 **Ingest Address** and port, and the token's value. You can find the **Ingest Address** on the top nav's **Network Settings** page, under the **Data Sources** tab.

Splunk HEC > Cribl Cloud endpoint

```
curl -k "https://default.main.<Your-Org-ID>.cribl.cloud:8088/services/collector" \
    -H "Authorization: <token_value>" \
    -d '{"event": "Goats are better than ponies."}{"event": "Goats are better climbe
```

With a Cribl.Cloud Enterprise plan, generalize the above URL's `default.main` substring to `<group-name>.<workspaceID>` 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 **Max active requests** 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 parser not found”

These errors can occur if there is a trailing slash in the URL path. For example, posting to `services/collector/` instead of `services/collector` can trigger these messages. Ensure that the URL path is correct by closely examining the reported `urlPath`.

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.

10.1.5.2. 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.

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)

Enabled defaults to `No`. When toggled to `Yes`:

Certificate name: 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. Defaults to `No`. When toggled to `Yes`:

- **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). Defaults to `Yes`.
- **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 this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

On Cribl-managed Cloud Workers (with certain plan/license tiers), 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 no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

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 Cribl Edge's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Edge's data processing engine.

Max buffer size: 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 per connection, not just per Worker Process – and this can dramatically expand memory usage.)

Commit frequency: The number of events to send downstream before committing that Stream has read them. Defaults to `42`.

Max file size: 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 Edge applies the default `1 MB`.

Max queue size: 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 **Max PQ size per Worker Process** in 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 Edge will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

As of Cribl Edge 4.1, Source-side PQ's default **Mode** changed from `Smart` to `Always On`. This option more reliably ensures events' delivery, and the change does not affect existing Sources' configurations. However:

- If you create Stream Sources programmatically, and you want to enforce the previous `Smart` mode, you'll need to update your existing code.
- If you enable `Always on`, this can reduce data throughput. As a trade-off for data durability, you might need to either accept slower throughput, or provision more machines/faster disks.
- You can optimize Workers' startup connections and CPU load at **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), maxiumum `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 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 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, unauthenticated 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 to `Yes` 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).

Max active connections: 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.

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: When set to `Yes`, this setting drops Splunk control fields such as `crcSalt` and `savedPort`. Defaults to `Yes`. Toggle to `No` if you want to retain these fields and store the data in the `__ctrlFields` internal field.

Extract metrics: Toggle to `Yes` 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.

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.

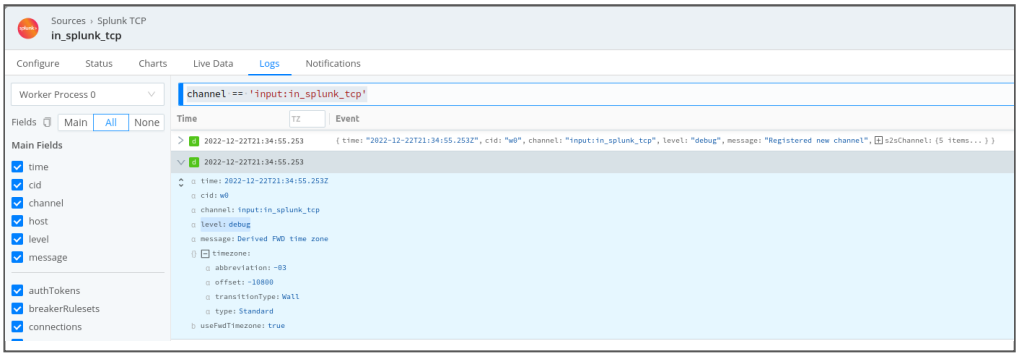
Using the UF Time Zone

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
```

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 t
# forceTimebasedAutoLB = false
```

With a Cribl.Cloud Enterprise plan, generalize the above URL's `default.main` substring to `<group-name>.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.

10.1.6. Windows

10.1.6.1. 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

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 to `Yes`, 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.

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 = `No`	Only new events generated after it receives the subscription are sent.
Read Existing Events = `Yes` & **Use Bookmarks** = `No`	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

Setting	Behavior
	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 = `Yes` & **Use Bookmarks** = `Yes`	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 `Yes`.

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`.

- **Name**: Field name.
- **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.

WEF subscriptions have a 22 `EventID` limit. To monitor more `EventIDs`, create multiple subscriptions.

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 TimeCreate
  </Query>
</QueryList>
```

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 this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

On Cribl-managed Cloud Workers (with certain plan/license tiers), 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 no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

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 Cribl Edge's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Edge's data processing engine.

Max buffer size: 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 per connection, not just per Worker Process – and this can dramatically expand memory usage.)

Commit frequency: The number of events to send downstream before committing that Cribl Edge has read them. Defaults to `42`.

Max file size: 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 on. If not specified, Cribl Edge applies the default `1 MB`.

Max queue size: 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 on. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in 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 Edge will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

In Cribl Edge 4.1 and later, Source-side PQ's default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Always On versus Smart Mode.

You can optimize Edge Nodes' startup connections and CPU load at **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.)

Name: Field name.

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 to process data from this input before the data is sent through the Routes.

Advanced Settings

Allow MachineID mismatch: Set this to `Yes` if you do not want to verify that the events sent by a client match the client's certificate Common Name (Subject CN). If set to `No`, 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 a `No` value.

Show originating IP: Toggle to `Yes` 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 this to `Yes` to add request headers to events, in the `__headers` field.

Max active requests: Maximum number of active requests allowed for this Source, per Worker Process. Defaults to `256`. Enter `0` for unlimited.

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.

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.

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.

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 set to `Yes`.
- `__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 set to `No`, the original client IP/port in this header will override the value of `__srcIpPort`.

If **Show originating IP** is set to `Yes`, 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.m
```

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**.

Subscription 22 EventID Limit

Issue: No events are forwarded. Windows Event Forwarding (WEF) subscriptions cannot include more than 22 `EventIDs` in a single filter.

Solution: To monitor more `EventIDs`, create multiple subscriptions, each with a filter containing no more than 22 `EventIDs`.

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

For more context and greater detail, see ATA Learning's WEF tutorial.

For more about WEF subscriptions, including sample subscription queries, see:

- Microsoft's Use Windows Event Forwarding to Help with Intrusion Detection topic.
- The NSA Cybersecurity Directorate's WEF Guidance.
- Palantir's WEF Guidance.

10.1.6.1.1. 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/
   ```

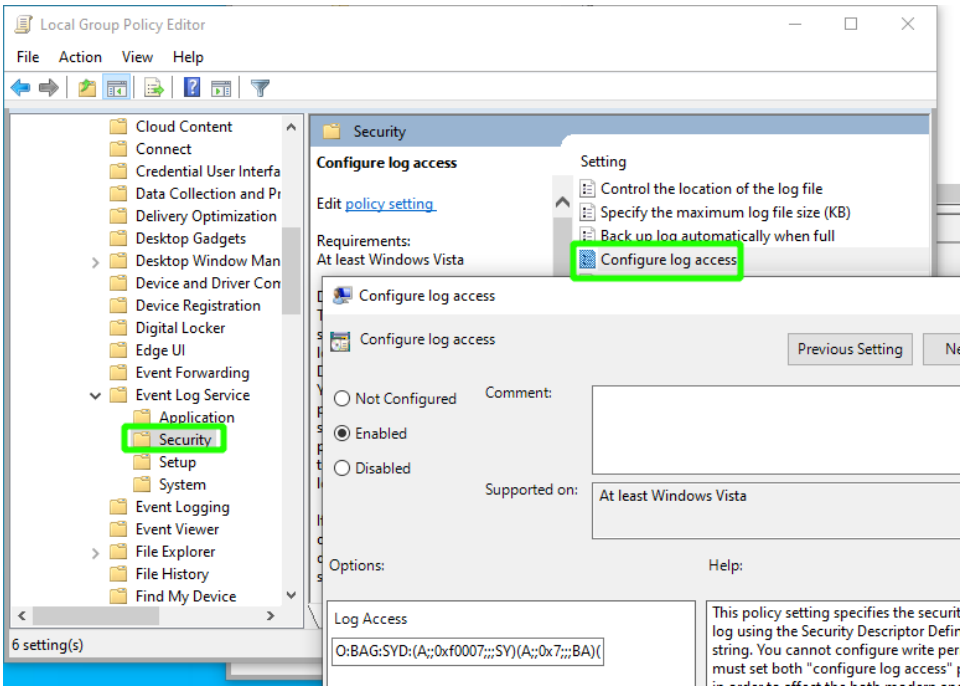
   Cribl.Cloud:

   ```
   https://<group-name>.main.<org-name>.cribl.cloud:<wef-source-port>/wsman/Subscri
   ```

   - 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-
```

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 set **Show originating IP** to `Yes` (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.

10.1.6.1.2. 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
```

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`.
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
```

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.

Now you can provide Kerberos authentication to a Windows Event Forwarder Source.

10.1.6.2. Windows Event Logs Source

Cribl Edge supports collecting local Windows Event Logs.

For guidance on workflow, see our better practices doc: Windows Observability Using Cribl Edge.

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.

For more information about how to locate Windows event logs on a Windows server, see Locate Windows Events Logs in the Server.

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**.
    - 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. `Security` is prefilled as a default. Click **Add log** to specify more event logs (for example, `Application` or `System`).
3. Next, you can configure the following **Optional Settings**:
    - **Read mode**: Select **Entire Log** (the default) to read all of the historical events and new events. Select **From last entry** to read only 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 to `Yes` to display rendered message strings in the XML and JSON event output.
- This toggle only appears when **Use Windows Tools** is set to `No`.
  - Defaults to `Yes` when JSON is the selected event format and `No` when XML is the event format.
  - Events will not contain a `Message` property when this is set to `No`.
- **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 add Fields to each event using Eval-like functionality.

Name: Field name.

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 to process data from this input before the data is sent through the Routes.

Advanced Settings

Use Windows Tools: Set to `No` by default.

- Set to `No`:
  - 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.
- Set to `Yes`:
    - The Windows Event Logs Source will collect events using external tools like `PS 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 set to `Yes`.

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 the **Use Windows Tools** toggle is set to `No` 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.
- 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.

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`.

10.1.6.3. 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.) Edge Leaders support configuring only one Windows Metrics Source per Fleet.

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 **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. Configure the following under **General Settings**:
    - **Input ID**: This is prefilled with the default value `in_windows_metrics`, which cannot be changed via the UI, due to the single-Source restriction above.
    - **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.

Adding a Process Set

To add a Process Set:

1. Open the Windows Metrics Source and access the **Configure** tab.
2. Click 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 children**: When toggled to “Yes”, 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, click **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 Filters 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.

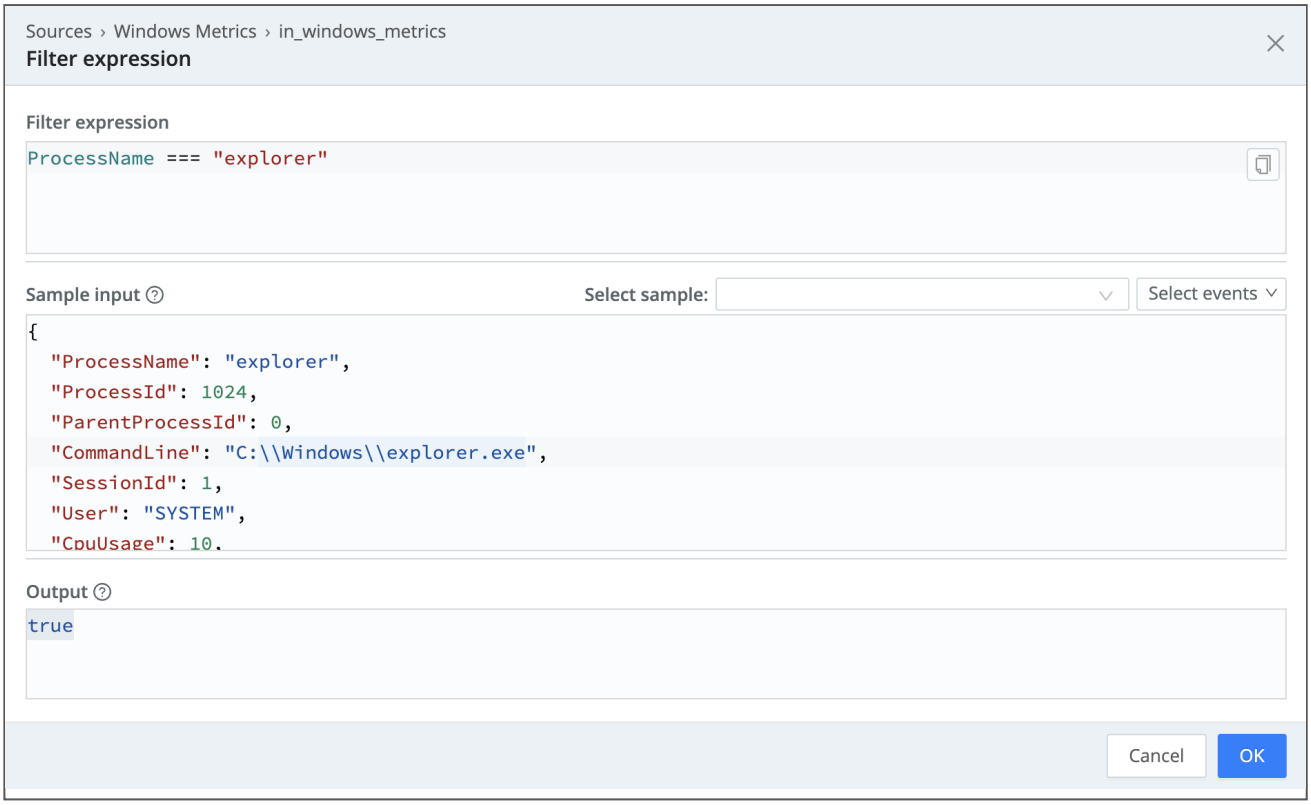
Supported Processes

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 add Fields to each event using Eval-like functionality.

Name: Field name.

Value: JavaScript expression to compute field's value, enclosed in quotes or backticks. (Can evaluate to a constant.)

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 set to `Yes`, 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: Set to `No` by default.

- Set to `No`: The Windows Metrics Source interacts directly with the Windows Management Instrumentation (WMI) API, resulting in faster metrics collection.
- Set to `Yes`: The Windows Metrics Source will collect events using `PS 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.

10.1.7. AppScope Source

AppScope is an open-source instrumentation utility from Cribl. It offers visibility into any Linux command or application, regardless of runtime, with no code modification. For details about configuring the AppScope CLI, loader, and library, see: https://appscope.dev/docs. Note that AppScope is no longer being actively developed by Cribl.

Type: **Push** | TLS Support: **YES** | Event Breaker Support: **YES**

Download AppScope

Cribl Edge must download the AppScope package from the Cribl CDN the first time it performs an operation on this Source. This will also update the AppScope package's version, if a newer one is available in the CDN.

If Cribl Edge cannot access the CDN (for example, if you are working on an airgapped instance), you should manually download the binary and place it in `$CRIBL_HOME/state/download/scope`.

Configure Cribl Edge to Receive AppScope 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 AppScope Source definition. If you clone this Source, Cribl Edge will add `-CLONE` to the original **Input ID**.
    - **Description**: Optionally, enter a description.
    - **UNIX domain socket**: For details, see Unix Domain Socket Fields below.
3. Under **Authentication**, select an **Authentication method** from the dropdown:
    - **Manual**: Use this default option to enter the shared secret clients must provide in the `authToken` header field. Select **Generate** if you need a new auth token. If empty, unauthenticated access will be permitted.

- **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. 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.

4. Next, you can configure the following **Optional Settings**:
   - **UNIX socket permissions**: Permissions to set for socket. For the preconfigured `in_appscope` source, defaults to `777`. When creating a new AppScope Source, you should set this to `777`. If empty, falls back to the runtime user's default permissions.
   - **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 TLS settings, AppScope Rules, Persistent Queue, Processing, Advanced settings and Connected Destinations outlined in the sections below.
6. Select **Save**, then **Commit & Deploy**.

Unix Domain Socket Fields

When set to `Yes`, the **Unix doman socket** field exposes the following two fields to specify a file-backed UNIX domain socket connection to listen on.

- **UNIX socket path**: Path to the UNIX domain socket. Defaults to `$CRIBL_HOME/state/appscope.sock`.
- **UNIX socket permissions**: Permissions to set for this socket, for example, `777`. If empty, Cribl Edge will use the runtime user's default permissions. (Appears under **Optional Settings**)

When **UNIX domain socket** is set to `No`, you instead see the following two fields to specify a network host and port.

- **Address**: Enter the hostname/IP on which to listen for AppScope data. (For example, `localhost`.) Defaults to `0.0.0.0`, meaning all addresses.
- **Port**: Enter the port number to listen on.

By default:

- In Cribl Stream, **UNIX domain socket** is set to `No`, with default network connections (address and port) of `0.0.0.0:10090` for TCP, and `0.0.0.0:10091` for TLS, respectively.
- In Cribl Edge, **UNIX domain socket** is set to `Yes`, with a UNIX socket path of `$CRIBL_HOME/state/appscope.sock`.

TLS Settings (Server Side)

This left tab is displayed only when the Optional Settings > **UNIX domain socket** toggle is set to `No`. It provides the following options.

Enabled defaults to `No`. When toggled to `Yes`:

Certificate name: 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. Defaults to `No`. When toggled to `Yes`:

- **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). Defaults to `Yes`.
- **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.

AppScope Rules

The AppScope Rules settings are available:

- In Cribl Stream single-instance – but not Cribl.Cloud – deployments.
- In Cribl Edge, both single-instance and Cloud.

Rules: Click **Add Rule** to include processes to scope, and to link to an AppScope config. Once you have saved the configuration, and committed and deployed your changes, AppScope will instrument any process that

matches a Rule, on any Edge Node in the Fleet.

(When no Rules are defined, you can still scope by PID in the Edge Processes page. Scope by PID only instruments a single process running in one Edge Node.)

- **Process name**: Matches if the string value you enter corresponds to the basename of the scoped process.
- **Process argument**: Matches if the string value you enter appears as a substring anywhere in the scoped process' full command line (including options and arguments).
- **AppScope config**: Select an AppScope config.

Transport override: Enter a URL to override aspects of the transport configuration, such as the hostname, port, or TLS settings. For details, see Transport Override Details.

Transport Override Details

In scenarios like the following, use the **Transport override** option to extend the defaults in AppScope's transport configuration:

- When this Source is set to **TCP** mode, it typically listens on the default address `0.0.0.0`. Scoped clients will need a specific `IP` or `hostname` to connect. In these cases, set **Transport override** URL to a specific IP/hostname (example format: `tcp://my.host.name`). This Source will parse the URL and look for the `hostname` and `port`, then use those values to override what would otherwise be sent to the `scope start` command.
- When this Source is set to **UNIX domain socket**, it listens by default on `$CRIBL_HOME/state/appscope.sock`. The socket is often created on a mounted volume in a container. The path to that socket might be different outside the container, or when mounted into another container. In these cases, set the **Transport override** URL to specify an alternative path (example format: `unix:///some/other/volume/appscope.sock`).

Persistent Queue Settings

In this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

On Cribl-managed Cloud Workers (with certain plan tiers), 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 no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a

> hybrid Group.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

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 Cribl Edge's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Edge's data processing engine.

Max buffer size: 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 per connection, not just per Worker Process – and this can dramatically expand memory usage.)

Commit frequency: The number of events to send downstream before committing that Stream has read them. Defaults to `42`.

Max file size: 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, etc. If not specified, Cribl Edge applies the default `1 MB`.

Max queue size: 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`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in 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 Edge will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

In Cribl Edge 4.1 and later, Source-side PQ's default **Mode** is `Always On`, to best ensure events' delivery. For details on optimizing this selection, see Always On versus Smart Mode.

You can optimize Workers' startup connections and CPU load at **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), maxiumum `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 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 to process data from this input before the data is sent through the Routes.

Disk Spooling

For Cribl Search to access the data that arrives at an AppScope Source, **Disk Spooling** must be enabled.

Enable disk persistence: Whether to save metrics to disk. When set to `Yes`, 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/appscope.sock`.

Advanced Settings

Enable proxy protocol: Toggle to `Yes` 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.

Max active connections: Maximum number of active connections allowed per Worker Process. Defaults to `1000`; enter `0` to allow unlimited connections.

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.

AppScope with Edge on Kubernetes

When Cribl Edge detects that a `scope`'d process is running inside a Kubernetes container, it reports the Kubernetes metadata as `kube_**` properties. It adds these to the incoming events and metrics, and you can view the combined events and metrics on the AppScope Source's **Live Data** tab.

For Cribl Edge to detect that it's running in a Kubernetes Pod, you must first set the `CRIBL_K8S_POD` environment variable.

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`
- `__srcIpPort`

Examples

The following examples work only in Cribl Stream. You can vary the scoped commands (`ps -ef` and `curl`) as desired.

Cribl.Cloud – TLS

An `in_appscope_tls` TLS Source is preconfigured for you on Cribl.Cloud, using port `10090`. You can send it AppScope data using this command:

```
./scope run -c <Your-Ingress-Address>:10090 -- ps -ef
```

Cribl Cloud – TCP

An `in_appscope_tcp` TCP Source is preconfigured for you on Cribl.Cloud, using port `10091`. You can send it AppScope data using this command:

```
./scope run -c tcp://<Your-Ingress-Address>:10091 -- curl -so /dev/null \
https://wttr.in/94105
```

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.

10.1.8. 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**

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.

Datadog Agent also emits Application Performance Monitoring data, which it sends to Datadog. Cribl Edge does not currently support ingesting this APM data.

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 `No`. When toggled to `Yes`:

Certificate name: 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. Defaults to `No`. When toggled to `Yes`:

- **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). Defaults to `Yes`.
- **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 this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

> On Cribl-managed Cloud Workers (with certain plan/license tiers), 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 no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

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 Cribl Edge's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Edge's data processing engine.

Max buffer size: 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 per connection, not just per Worker Process – and this can dramatically expand memory usage.)

Commit frequency: The number of events to send downstream before committing that Stream has read them. Defaults to `42`.

Max file size: 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, etc. If not specified, Cribl Edge applies the default `1 MB`.

Max queue size: 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`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in 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 Edge will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

In Cribl Edge 4.1 and later, Source-side PQ's default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Always On versus Smart Mode.

You can optimize Workers' startup connections and CPU load at **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 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 to process data from this input before the data is sent through the Routes.

Advanced Settings

Show originating IP: Toggle to `Yes` 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 this to `Yes` to add request headers to events, in the `__headers` field.

Health check endpoint: Toggle to `Yes` 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.

Max active requests: Maximum number of active requests per worker process. Use `0` for unlimited.

Max requests per socket: 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 to `Yes` 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 to `No` (the default).

Forward API key validation requests: Toggle to `Yes` to send key validation requests from Datadog Agent to the Datadog API. If toggled to `No` (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). Defaults to `Yes`. Available when **Forward API key validation requests** is toggled to `Yes`.

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

Max requests per socket 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 **Max requests per socket** 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 **Max requests per socket**, 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 **Max requests per socket**.

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 set to `Yes`.
- `__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 set to `No`, the original client IP/port in this header will override the value of `__srcIpPort`.

If **Show originating IP** is set to `Yes`, 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
```

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`
APM Data	Can only be sent to Datadog, for example, `https://trace.agent.datadoghq.com`. Cannot be sent to Cribl Edge even when you **are** sending other types there.	`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** to `Yes`.
- 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.

10.1.9. 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.

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 to `Yes` 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)

Enabled defaults to `No`. When toggled to `Yes`:

Certificate name: 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. Defaults to `No`. When toggled to `Yes`:

- **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). Defaults to `Yes`.
- **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 this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

> On Cribl-managed Cloud Workers (with certain plan/license tiers), 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 no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

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 Cribl Edge's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Edge's data processing engine.

Max buffer size: 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 per connection, not just per Worker Process – and this can dramatically expand memory usage.)

Commit frequency: The number of events to send downstream before committing that Stream has read them. Defaults to `42`.

Max file size: 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, etc. If not specified, Cribl Edge applies the default `1 MB`.

Max queue size: 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`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in 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 Edge will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

In Cribl Edge 4.1 and later, Source-side PQ's default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Always On versus Smart Mode.

You can optimize Workers' startup connections and CPU load at **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 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 to process data from this input before the data is sent through the Routes.

Advanced Settings

Show originating IP: Toggle to `Yes` 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 this to `Yes` to add request headers to events, in the `__headers` field.

Health check endpoint: Toggle to `Yes` 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.

Max active requests: 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.

Max requests per socket: 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 you toggle this to `Yes`, 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. Sliding **Enable proxy mode** to `Yes` exposes 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). Defaults to `Yes`.

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

Max requests per socket 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 **Max requests per socket** 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 **Max requests per socket**, 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 **Max requests per socket**.

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

The Elasticsearch API input normalizes the following fields:

- `@timestamp` becomes `_time` at millisecond resolution.
- `host` is set to `host.name`.
- Original object `host` is stored in `__host`.

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 set to `Yes`.
- `__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 set to `No`, the original client IP/port in this header will override the value of `__srcIpPort`.

If **Show originating IP** is set to `Yes`, 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"
```

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.

10.1.10. 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**:
    - **Auth tokens**: Shared secrets to be provided by any client (`Authorization: <token>`). Click **Generate** to create a new secret. If empty, unauthenticated 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 below.
    - **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`. See Splunk HEC Details below.
    - **Splunk HEC Acks**: Whether to enable Splunk HEC acknowledgements. Defaults to `No`.
    - **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 Details

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.)

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)

Enabled defaults to `No`. When toggled to `Yes`:

Certificate name: 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. Defaults to `No`. When toggled to `Yes`:

- **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). Defaults to `Yes`.
- **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 this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

> On Cribl-managed Cloud Workers (with certain plan/license tiers), 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 no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

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 Cribl Edge's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Edge's data processing engine.

Max buffer size: 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 per connection, not just per Worker Process – and this can dramatically expand memory usage.)

Commit frequency: The number of events to send downstream before committing that Stream has read them. Defaults to `42`.

Max file size: 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 Edge applies the default `1 MB`.

Max queue size: 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 **Max PQ size per Worker Process** in 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 Edge will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

In Cribl Edge 4.1 and later, Source-side PQ's default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Always On versus Smart Mode.

You can optimize Workers' startup connections and CPU load at **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 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 to process data from this input before the data is sent through the Routes.

Advanced Settings

Show originating IP: Toggle to `Yes` 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 this to `Yes` to add request headers to events, in the `__headers` field.

Health check endpoint: Toggle to `Yes` 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.

Max active requests: 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; and so forth.

Max requests per socket: 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

Max requests per socket 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 **Max requests per socket** 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 **Max requests per socket**, 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 **Max requests per socket**.

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 set to `Yes`.
- `__inputId`
- `__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 set to `No`, the original client IP/port in this header will override the value of `__srcIpPort`.

If **Show originating IP** is set to `Yes`, 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":"my
{"_time":1541280341, "host":"myOtherHost", "source":"myOtherSource", "_raw": "{\"mes
```

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

Cribl Edge

The examples in this section demonstrate sending HTTP data into a Cribl Edge binary that you manage on-prem, or on a VM. To set up these examples:

1. Configure Cribl to listen on port `10080` for HTTP (default). Set `authToken` to `myToken42`.
2. Send a payload to your Cribl Edge receiver.

Cribl Endpoint – Single Event

Cribl Single Event Example:

```
curl -k http://<myCriblHost>:10080/cribl/_bulk -H 'Authorization: myToken42' -d '{"_
```

Cribl Endpoint – Multiple Events

Cribl Endpoint - Multiple Events

```
curl -k http://<myCriblHost>:10080/cribl/_bulk -H 'Authorization: myToken42' -d $'{"
```

Splunk HEC Event Endpoint

Splunk HEC Event Endpoint

```
curl -k http://<myCriblHost>:10080/services/collector/event -H 'Authorization: myTok

curl -k http://<myCriblHost>:10080/services/collector -H 'Authorization: myToken42' -
```

For Splunk HEC, the token specification can be either `Splunk <token>` or `<token>`.

Cribl.Cloud – Single Event

1. Generate and copy a token in your Cribl.Cloud instance's HTTP Source > **General Settings**.
2. From the command line, use `https`, your Cribl.Cloud portal's **Ingest Address** and port, and the token's value. You can find the **Ingest Address** on the top nav's **Network Settings** page, under the **Data Sources** tab.

Cribl.Cloud – Single Event

```
curl -k https://default.main.<Your-Org-ID>.cribl.cloud:10080/cribl/_bulk -H 'Authori
```

With a Cribl.Cloud Enterprise plan, generalize the above URL's `default.main` substring to `<group-name>.main` when sending to other Fleets.

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.

10.1.11. 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.

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.
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**.

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)

Enabled defaults to `No`. When toggled to `Yes`:

Certificate name: 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. Defaults to `No`. When toggled to `Yes`:

- **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). Defaults to `Yes`.
- **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 this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

> On Cribl-managed Cloud Workers (with certain plan/license tiers), 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 no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

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 Cribl Edge's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Edge's data processing engine.

Max buffer size: 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 per connection, not just per Worker Process – and this can dramatically expand memory usage.)

Commit frequency: The number of events to send downstream before committing that Stream has read them. Defaults to `42`.

Max file size: 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, etc. If not specified, Cribl Edge applies the default `1 MB`.

Max queue size: 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`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in 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 Edge will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

In Cribl Edge 4.1 and later, Source-side PQ's default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Always On versus Smart Mode.

You can optimize Workers' startup connections and CPU load at **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), maxiumum `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 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 to process data from this input before the data is sent through the Routes.

Advanced Settings

Show originating IP: Toggle to `Yes` 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 this to `Yes` to add request headers to events, in the `__headers` field.

Health check endpoint: Toggle to `Yes` 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.

Max active requests: 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.

Max requests per socket: 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

Max requests per socket 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 **Max requests per socket** 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 **Max requests per socket**, 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 **Max requests per socket**.

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 set to `No`, the original client IP/port in this header will override the value of `__srcIpPort`.

If **Show originating IP** is set to `Yes`, 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.

10.1.12. 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 `No`. When toggled to `Yes`:

Certificate name: 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. Defaults to `No`. When toggled to `Yes`:

- **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). Defaults to `Yes`.
- **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 this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

On Cribl-managed Cloud Workers (with certain plan/license tiers), 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 no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

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 Cribl Edge's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Edge's data processing engine.

Max buffer size: 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 per connection, not just per Worker Process – and this can dramatically expand memory usage.)

If you do not see the **Max buffer size** setting, it is because you enabled the Persistent Queue. The Persistent Queue provides its own configurable internal buffer, and **Max buffer size** is not applicable in that case.

Commit frequency: The number of events to send downstream before committing that Stream has read them. Defaults to `42`.

Max file size: 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, etc. If not specified, Cribl Edge applies the default `1 MB`.

Max queue size: 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`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in 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 Edge will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

In Cribl Edge 4.1 and later, Source-side PQ's default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Always On versus Smart Mode.

You can optimize Workers' startup connections and CPU load at **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 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 to process data from this input before the data is sent through the Routes.

Advanced Settings

Enable Proxy Protocol: Toggle to `Yes` 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.)

**Max buffer size (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
....
```

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
```

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
```

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
```

Graphite Example without Dimensions

```
metric1 100 9999
metric2 200 9999
metric.dot.3 300.16 9999.16
```

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.

10.1.13. 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 `No`. When toggled to `Yes`:

Certificate name: Name of the predefined certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

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. Defaults to `No`. When toggled to `Yes`:

- **Validate client 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 `Yes`.
- **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 this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

> On Cribl-managed Cloud Workers (with certain plan/license tiers), 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 of disk space allocated per PQ-enabled Source, per Worker Process.
>
> The 1 GB limit is on uncompressed inbound data, and no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

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 Cribl Edge's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Edge's data processing engine.

Max buffer size: 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 per connection, not just per Worker Process – and this can dramatically expand memory usage.)

Commit frequency: The number of events to send downstream before committing that Cribl Edge has read them. Defaults to `42`.

Max file size: 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 Edge applies the default `1 MB`.

Max queue size: 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 **Max PQ size per Worker Process** in 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 Edge will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

In Cribl Edge 4.1 and later, Source-side PQ's default **Mode** is `Always on`. This setting helps ensure events the best chance of delivery. For details on optimizing this selection, see Always On versus Smart Mode.

You can optimize Worker startup connections and CPU load at **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 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 to process data from this input before the data is sent through the Routes.

Advanced Settings

Max active connections: 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.

10.1.14. NetFlow Source

Cribl Edge supports receiving NetFlow v5 and v9 data via UDP.

Type: **Push** | TLS Support: **No** | Event Breaker Support: **No**

This Source ingests NetFlow records similarly to how it ingests events from other upstream senders: fields are broken out, and each record includes a message header. If you prefer to render NetFlow data as metrics, use a pre-processing Pipeline or a Route.

If you want to associate a Community ID with a network flow that passes through this Source, connect the Source to a Route containing a Function that supports expression evaluation; then, in the Function, use the C.Net.communityIDv1() method. You can even generate different CommunityIDs for different use cases within the same network flow – to do this, use multiple `communityIDv1()` methods, with the optional `seed` parameter.

Configure Cribl Edge to Receive NetFlow 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 NetFlow Source definition.
    - **Description**: Optionally, enter a description.
    - **Address**: Enter the hostname/IP to listen for NetFlow data. For example: `localhost`, `0.0.0.0`, or `::`.
    - **Port**: Enter the port number.
    - **Enable pass-through:** Enable this setting to allow Cribl Edge to forward events to a NetFlow Destination. If enabled, Cribl Edge generates an additional event containing `__netflowRaw`, which directly routes the raw NetFlow packet data to a downstream NetFlow Destination. These additional events do not count against any ingest quotas.
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**.

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

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 this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

On Cribl-managed Cloud Workers (with certain plan/license tiers), 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 no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

For more about PQ modes, see Always On versus Smart Mode.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

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 Cribl Edge's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Edge's data processing engine.

Max buffer size: 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 per connection, not just per

Worker Process – and this can dramatically expand memory usage.)

Commit frequency: The number of events to send downstream before committing that Cribl Edge has read them. Defaults to `42`.

Max file size: 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, etc. If not specified, Cribl Edge applies the default `1 MB`.

Max queue size: 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`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in 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 Edge will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

You can optimize Workers' startup connections and CPU load at **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 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 to process data from this input before the data is sent through the Routes.

Advanced Settings

IP allowlist regex: Regex matching IP addresses that are allowed to establish a connection. Defaults to `.*` (that is, all IPs).

IP denylist regex: Regex matching IP addresses whose messages you want this Source to ignore. Defaults to `^$` (that is, every specific IP address in the list). This takes precedence over the allowlist.

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. Minimum: `256`. Maximum: `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.

Template cache minutes (minutes): This setting only affects v9 flow records. Use this setting to specify the length of time (in minutes) that Cribl Edge caches NetFlow v9 templates before discarding them. For example, setting the duration to `30` means Cribl Edge will cache templates for 30 minutes. If flow records are not refreshed by a new event, the templates are discarded.

The template cache minutes setting helps optimize the performance and reliability of NetFlow v9 data processing. This setting manages memory usage and ensures that outdated templates do not persist indefinitely. As you tune your NetFlow source, the goal is to balance system performance against memory use. Monitor your network and adjust this setting to find the right balance between template availability and system data overload. Some considerations when tuning your system:

- If your network's data patterns change often, consider using a shorter cache duration. Conversely, if the data records are stable and change less frequently, try using a longer duration.
- Monitor the status of this Source across your Worker/Edge nodes. If you notice Cribl Edge discards FlowSets, it may mean the cache duration is too short. Try increasing the cache duration and then observer whether there is a performance improvement. Alternatively, adjust your NetFlow exporters to send templates more frequently.
- If your system is using too much memory, it may mean that your cache duration is too long, especially if you have a lot of NetFlow exporters sending data to a single Source.

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 NetFlow 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. We have organized these fields into categories to make them easier to grasp; the categories are our own, and not from the NetFlow specifications. The field definitions are mostly copied from Cisco NetFlow documentation.

The fields for NetFlow v5 and v9 data are different.

Fields for NetFlow v9

For NetFlow v9, Cisco-registered fields map to their most natural representation in Cribl. Specifically:

- IPv4 addresses are represented as dotted octet strings.
- IPv6 as colon-separated octet strings.
- MAC addresses also as colon-separated strings.
- Numbers that are six bytes or less are represented as numbers.
- Numbers that are greater than six bytes are represented as strings.
- Registered field attribute names are the camelCase versions of the SCREAMING_CASE names in the Cisco whitepaper.

Cribl Edge collects and offers unregistered fields in an array of records consisting of their type codes and byte buffers. Cribl Edge can accept and handle additional Internet Protocol Flow Information Export (IPFIX) codes. However, Cribl Edge does not decode those values. You can add Pipeline functions to change how these fields are processed by default.

Not all NetFlow v9 fields will appear on all events. Only those specified by the template for that specific data record. These fields are available for NetFlow v9:

- `inBytes`: The number of incoming Bytes.
- `inPkts`: The number of incoming packets.
- `flows`: The number of flows that were aggregated.
- `protocol`: The IP protocol.
- `srcTos`: The type of service setting when entering the incoming interface.
- `tcpFlags`: The accumulation of all the TCP flags seen in this flow.

- `l4SrcPort`: The TCP/UDP source port number.
- `ipv4SrcAddr`: The IPv4 source address.
- `srcMask`: The number of contiguous bits in the source subnet mask.
- `inputSnmp`: The input interface index.
- `l4DstPort`: The TCP/UDP destination port number.
- `ipv4DstAddr`: The IPv4 destination address.
- `dstMask`: The number of contiguous bits in the destination subnet mask.
- `outputSnmp`: The output interface index.
- `ipv4NextHop`: The IPv4 address of the next-hop router.
- `srcAs`: The source BGP autonomous system number. The length may be and defaults to 2.
- `dstAs`: The destination BGP autonomous system number. The length may be and defaults to 2.
- `bgpIpv4NextHop`: The IPv4 address of the next-hop router in the bgp domain.
- `mulDstPkts`: The number of IP multicast outgoing packets.
- `mulDstBytes`: The number of IP multicast outgoing bytes.
- `lastSwitched`: The system uptime in milliseconds at which the last packet of this flow was switched.
- `firstSwitched`: The system uptime in milliseconds at which the first package of this flow was switched.
- `outBytes`: The number of outgoing bytes.
- `outPkts`: The number of outgoing packets.
- `minPktLngth`: The minimum IP packet length on incoming packets.
- `maxPktLngth`: The maximum IP packet length on incoming packets.
- `ipv6SrcAddr`: The IPv6 source address.
- `ipv6DstAddr`: The IPv6 destination address.
- `ipv6SrcMask`: The number of contiguous bits in the IPv6 source mask.
- `ipv6DstMask`: The number of contiguous bits in the IPv6 destination mask.
- `ipv6FlowLabel`: The IPv6 flow label as per the RFC 2460 definition.
- `icmpType`: The ICMP packet type, reported as type * 256 + code.
- `mulIgmpType`: The IGMP packet type.
- `samplingInterval`: The rate at which packets are sampled; a value of 100 indicates that 1% of packets are sampled.
- `samplingAlgorithm`: The platform sampling algorithm; 1 is deterministic, 2 is random.
- `flowActiveTimeout`: Timeout value in seconds for active flow entries in the NetFlow cache.
- `flowInactiveTimeout`: Timeout value in seconds for inactive flow entries in the NetFlow cache.

- `engineType`: The type of flow switching engine; 0 is RP, 1 is VIP/Linecard.
- `engineId`: The ID number of the flow switching engine.
- `totalBytesExp`: The number of bytes exported by the observation domain.
- `totalPktsExp`: The number of packets exported by the observation domain.
- `totalFlowsExp`: The number of flows exported by the observation domain.
- `ipv4SrcPrefix`: The IPv4 source address prefix, specific for Catalyst architecture.
- `ipv4DstPrefix`: The IPv4 destination address prefix, specific for Catalyst architecture.
- `mplsTopLabelType`: The MPLS top label type; 0 is unknown, 1 is TE midpoint, 2 is AToM, 3 is VPN, 4 is BGP, and 5 is LDP.
- `mplsTopLabelIpAddr`: The IPv4 address of the forwarding equivalent class corresponding to the MPLS top label.
- `flowSamplerId`: The ID number of the flow sampler.
- `flowSampleMode`: The type of algorithm used for sample data.
- `flowSamplerRandomInterval`: The packet interval at which to sample.
- `minTtl`: Minimum TTL on incoming packets.
- `maxTtl`: Maximum TTL on incoming packets.
- `ipv4Ident`: The IPv4 identification field.
- `dstTos`: The type of service byte setting when exiting the outgoing interface.
- `inSrcMac`: The incoming source MAC address.
- `outDstMac`: The outgoing destination MAC address.
- `srcVlan`: The virtual LAN ID associated with the ingress interface.
- `dstVlan`: The virtual LAN ID associated with the egree interface.
- `ipProtocolVersion`: The IP version. The default is assumed to be 4 if unspecified.
- `direction`: The flow direction; 0 is ingress, 1 is egress.
- `ipv6NextHop`: The IPv6 address of the next-hop router.
- `bgpIpv6NextHop`: The IPv6 address of the next-hop router in the bgp domain.
- `ipv6OptionHeaders`: The IPv6 option headers found in the flow.
- `mplsLabel1`: The MPLS label at position 1 in the stack.
- `mplsLabel2`: The MPLS label at position 2 in the stack.
- `mplsLabel3`: The MPLS label at position 3 in the stack.
- `mplsLabel4`: The MPLS label at position 4 in the stack.
- `mplsLabel5`: The MPLS label at position 5 in the stack.

- `mplsLabel6`: The MPLS label at position 6 in the stack.
- `mplsLabel7`: The MPLS label at position 7 in the stack.
- `mplsLabel8`: The MPLS label at position 8 in the stack.
- `mplsLabel9`: The MPLS label at position 9 in the stack.
- `mplsLabel10`: The MPLS label at position 10 in the stack.
- `inDstMac`: The incoming destination MAC address.
- `outSrcMac`: The outgoing source MAC address.
- `ifName`: The shortened interface name.
- `ifDesc`: The full interface name.
- `samplerName`: The name of the flow sampler.
- `inPermanentBytes`: The running byte counter for a permanent flow.
- `inPermanentPkts`: The running packet counter for a permanent flow.
- `fragmentOffset`: The fragment-offset value from fragmented IP packets.
- `forwardingStatus`: The forwarding status.
- `mplsPalRd`: The MPLS pal route distinguisher.
- `mplsPrefixLen`: The number of consecutive bits in the MPLS prefix length.
- `srcTrafficIndex`: The BGP policy accounting source traffic index.
- `dstTrafficIndex`: The BGP policy accounting destination traffic index.
- `applicationDescription`: The application description.
- `applicationTag`: The application tag.
- `applicationName`: The name associated with a classification.
- `postIpDiffServeCodePoint`: The value of a differentiated services code point encoded in the differentiated services field, after modification.
- `replicationFactor`: The multicast replication factor.
- `layer2PacketSectionOffset`: The layer 2 packet section offset, potentially a generic offset.
- `layer2PacketSectionSize`: The layer 2 packet section size, potentially a generic size.
- `layer2PacketSectionData`: The layer 2 packet section data.

Option Data Records for NetFlow v9

Options data records report aggregate metadata about the Exporters. Like data records, they can include any or all of the previous fields. Additionally, they also have a `scopes` field with a value containing an array of records with a `type` field. The valid types are:

- `system`
- `interface`
- `line-card`
- `cache`
- `template`

Option data records also have a `value` field, which is a natural number. The JavaScript type is either a number or a string, depending on the size of the value.

Fields for NetFlow v5

General

- `_time`: The UNIX timestamp (in 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.
- `inputInt`: SNMP index of input interface; always set to zero.
- `outputInt`: SNMP index of output interface.
- `protocol`: IP protocol type (for example, TCP = 6; UDP = 17) of the observed network flow.
- `tcpFlags`: Cumulative logical OR of TCP flags in the observed network flow.
- `tos`: IP type of service; switch sets it to the ToS of the first packet of the flow.
- `icmpType`: Type of the ICMP message. Only present if the protocol is ICMP.
- `icmpCode`: Code of the ICMP message. Only present if the protocol is ICMP.

Because this Source reads input data directly from bytes in a compact format, there is nothing suitable to put in a `_raw` field, and the Source does not add a `_raw` field to events.

Flow Source and Destination

- `srcAddr`: Source IP address; in case of destination-only flows, set to zero.
- `dstAddr`: Destination IP address.
- `nextHop`: IP address of next hop router.
- `srcPort`: TCP/UDP source port number.

- `dstPort`: TCP/UDP destination port number. If this is an ICMP flow, this field is a combination of ICMP type and ICMP code, which are broken out separately as `icmpType` and `icmpCode` fields.
- `srcMask`: Source address prefix mask bits.
- `dstMask`: Destination address prefix mask bits.
- `srcAs`: Autonomous system number of the source, either origin or peer.
- `dstAs`: Autonomous system number of the destination, either origin or peer.

Flow Statistics

- `packets`: Packets in the flow.
- `octets`: Total number of Layer 3 bytes in the packets of the flow.
- `startTime`: System uptime, in milliseconds, at the start of the flow.
- `endTime`: System uptime, in milliseconds, at the time the last packet of the flow was received.
- `durationMs`: `endTime` minus `startTime`.

Header

The following are subfields within the `header` field:

- `count`: Number of flows exported in this flow frame (protocol data unit, or PDU).
- `sysUptimeMs`: Current time in milliseconds since the export device booted.
- `unixSecs`: Current seconds since 0000 UTC 1970.
- `unixNsecs`: Residual nanoseconds since 0000 UTC 1970.
- `flowSequence`: Sequence counter of total flows seen.
- `engineType`: Type of flow switching engine.
- `engineId`: ID number of the flow switching engine.
- `samplingMode`: The first two bits of what Cisco NetFlow documentation calls `SAMPLING_INTERVAL`. These bits specify a sampling mode.
- `samplingInterval`: The remaining 14 bits of what Cisco NetFlow documentation calls `SAMPLING_INTERVAL`. These bits specify a sampling interval.
- `version`: NetFlow export format version number.

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:

- `__netflowRaw` : Directly routes the raw NetFlow packet data to a downstream NetFlow Destination.

Considerations for Working with NetFlow Data FlowSets

For both NetFlow v5 and v9, Cribl Edge:

- Can forward NetFlow packets to other NetFlow destinations. However, it cannot modify the the contents of the incoming packet. In other words, Cribl Edge forwards the packets verbatim as they came in.
- Only routes NetFlow packets from upstream exporters and cannot generate its own NetFlow packets.
- Cannot forward non-NetFlow input data to NetFlow Destinations.
- Can forward NetFlow packets to non-NetFlow Destinations, such as Splunk, Syslog, S3, and others. To send NetFlow flow records to other non-NetFlow Destinations, use the other events generated on the NetFlow Source that do not include the `__netflowRaw` field. NetFlow Destinations discard those events.

Cribl Edge emits:

- Decoded Cribl events downstream for each data record and options data record.
- When you enable pass-through mode: an undecoded Cribl event for the entire packet.

High Availability and Disaster Recovery for NetFlow v9

NetFlow v9 Template Records are sent every 20 packets or every 30 minutes. Cribl Edge uses a key-value store to share these records across Edge Nodes in a Fleet, ensuring availability for processing FlowSets.

- **Template Sharing**: Records are shared across Edge Nodes with a minimal delay, ensuring timely processing. The brief interval between receiving and sharing Template Records is designed to be negligible, given the low rate of Template Record changes.
- **Worker Coordination**: The key-value store allows uninterrupted processing during Edge Node failures.
- **Disaster Recovery**: Replication in the key-value store ensures Template Records are preserved, enabling seamless recovery.

UDP Tuning

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 may be too small.

You can check the current buffer size with:

```
$ sysctl net.core.rmem_max
```

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
```

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 (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
```

For the permanent settings change, add the following lines to `/etc/sysctl.conf`:

```
net.core.rmem_max=26214400
net.core.rmem_default=26214400
```

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.

Data Collisions

When Cribl Edge receives a NetFlow v9 template, it identifies the template using three values:

- Template ID
- Source ID, which is unique to the upstream exporting device
- IP address of the upstream exporting device

Cribl Edge uses these three values to prevent data collisions. However, data collisions may occur under these two conditions:

- Your system has a heterogeneous variety of upstream exporter devices, AND
- The Exporter IP addresses are rewritten (such as by a proxy server) before they are sent downstream to Cribl Edge.

In that situation, it is possible for two exporter devices with the same IP address to send packets simultaneously and cause a collision. The workaround is to create unique IP addresses for the exporters or use the original IP addresses.

10.1.15. OpenTelemetry (OTel) Source

Cribl Edge supports receiving trace, metric, and log events from OTLP-compliant senders. Cribl Edge supports log events beginning in v.4.7.

Type: **Push** | TLS Support: **YES** | Event Breaker Support: **No**

Supported and Unsupported Input Data

In Cribl Edge 4.0.3 and later, the Open Telemetry Source supports receiving compressed inbound data (with DEFLATE or gzip compression), as well as uncompressed data.

In Cribl Edge 4.1 and later, this Source supports receiving 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 currently does not support JSON Protobuf.

The OpenTelemetry Project's Data Sources 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.
    - **Address**: Enter the hostname/IP to listen on. Defaults to `0.0.0.0` (all addresses, IPv4 format).
    - **Port**: By default, OTel applications send output to port `4317` when using the gRPC protocol, and port `4318` when using HTTP. This setting defaults to `4317` – you must change it if you set **Protocol** (below) to `HTTP`, or you want Cribl Edge to collect data from an OTel application that is using a different port.

Port `4318` is not available on Cribl-managed Worker Groups in Cribl.Cloud.

3. Next, you can configure the following **Optional Settings**:
    - **Protocol**: Use the drop-down to choose the protocol matching the data you will ingest: `gRPC` (the default), or `HTTP`.
    - **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**: 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.

Enabled defaults to `No`. When toggled to `Yes`:

Certificate name: Name of the predefined certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

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). Defaults to `No`. When toggled to `Yes`:

- **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). Defaults to `Yes`.
- **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 this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

> On Cribl-managed Cloud Workers (with certain plan/license tiers), 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 no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

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 Cribl Edge's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Edge's data processing engine.

Max buffer size: 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 per connection, not just per Worker Process – and this can dramatically expand memory usage.)

Commit frequency: The number of events to send downstream before committing that Stream has read them. Defaults to `42`.

Max file size: 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, etc. If not specified, Cribl Edge applies the default `1 MB`.

Max queue size: 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`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in 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 Edge will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

In Cribl Edge 4.1 and later, Source-side PQ's default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Always On versus Smart Mode.

You can optimize Workers' startup connections and CPU load at **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 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 to process data from this input before the data is sent through the Routes.

Advanced Settings

Advanced Settings always displays **OTLP version**, **Extract spans**, **Extract metrics**, and **Environment**. When **OTLP version** is set to `1.3.1`, the UI also displays **Extract logs**.

The **Extract spans**, **Extract metrics**, and **Extract logs** settings are unique to OTel. Their default `No` settings allow Cribl Edge to essentially function as a bump on the wire, generating a single event for each incoming OTel event. This can be useful when, for example, you want to send whole OTel events to persistent storage.

- **OTLP version**: The drop-down offers `0.10.0` (default) and `1.3.1`.
- **Extract spans**: Toggle to `Yes` if you want Cribl Edge to generate an individual event for each span. (Recall that traces contain multiple spans.)
- **Extract metrics**: Toggle to `Yes` if you want Cribl Edge to generate an individual event for each data point. (Recall that OTel metric events contain multiple data points.)
- **Extract logs**: Available only when **OTLP version** is set to `1.3.1`. Toggle to `Yes` if you want Cribl Edge to generate an individual event for each log record. Cribl recommends doing this to make it easier to transform and manipulate logs.

- **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 `gRPC`, the UI displays one additional setting:

- **Max active connections**: 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.

When **General Settings** > **Protocol** is set to `HTTP`, the UI displays these additional settings:

- **Health check endpoint**: Toggle to `Yes` 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.
- **Max active requests**: Maximum number of active requests allowed for this Source, per Worker Process. Defaults to `256`. Enter `0` for unlimited.
- **Max requests per socket**: 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).

Balancing Connection Reuse Against Request Distribution

Max requests per socket 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 **Max requests per socket** 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 **Max requests per socket**, 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 **Max requests per socket**.

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.

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
   ```

   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** to `Yes`, 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 to `No`.

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 default.
   - Set `tls` > `insecure` to `false`. This matches your setting **TLS Settings (Server Side)** > **Enabled** to `Yes` on 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 `No` on 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
   ```

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.

10.1.16. 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 to `Yes` to enable SNMPv3 authentication and reveal authentication parameters. Defaults to `No`.

Allow unmatched traps: Toggle to `Yes` 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 to `No` (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 this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

On Cribl-managed Cloud Workers (with certain plan/license tiers), 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 no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

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 Cribl Edge's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Edge's data processing engine.

Max buffer size: 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 per connection, not just per Worker Process – and this can dramatically expand memory usage.)

> If you do not see the **Max buffer size** setting, it is because you enabled the Persistent Queue. The Persistent Queue provides its own configurable internal buffer, and **Max buffer size** is not applicable in that case.

Commit frequency: The number of events to send downstream before committing that Stream has read them. Defaults to `42`.

Max file size: 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 on. If not specified, Cribl Edge applies the default `1 MB`.

Max queue size: 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 on. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in 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 Edge will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

> In Cribl Edge 4.1 and later, Source-side PQ's default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Always On versus Smart Mode.

You can optimize Workers' startup connections and CPU load at **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 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 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.

Max buffer size (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.

10.1.17. 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.

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**.
- **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 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**.

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)

Enabled defaults to `No`. When toggled to `Yes`:

Certificate name: 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. Defaults to `No`. When toggled to `Yes`:

- **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). Defaults to `Yes`.
- **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 this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

> On Cribl-managed Cloud Workers (with certain plan/license tiers), 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 no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

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 Cribl Edge's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Edge's data processing engine.

Max buffer size: 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 per connection, not just per Worker Process – and this can dramatically expand memory usage.)

If you do not see the **Max buffer size** setting, it is because you enabled the Persistent Queue. The Persistent Queue provides its own configurable internal buffer, and **Max buffer size** is not applicable in that case.

Commit frequency: The number of events to send downstream before committing that Stream has read them. Defaults to `42`.

Max file size: 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, etc. If not specified, Cribl Edge applies the default `1 MB`.

Max queue size: 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`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in 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 Edge will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

In Cribl Edge 4.1 and later, Source-side PQ's default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Always On versus Smart Mode.

You can optimize Workers' startup connections and CPU load at **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 add Fields to each event, using Eval-like functionality.

Name: Field name.

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 to process data from this input before the data is sent through the Routes.

Advanced Settings

Enable Proxy Protocol: Toggle to `Yes` if the connection is proxied by a device that supports Proxy Protocol v1 or v2.

Single msg per UDP: Enable this to treat received UDP packet data as a full syslog message. With the `No` default, Cribl Edge will treat newlines within the packet as event delimiters.

Allow non-standard app name: Toggle to `Yes` 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 to `Yes` if you want the Source to load balance traffic across all Worker Processes, as explained below. (This setting is available only in Cribl Stream deployed in Distributed mode.)

IP allowlist regex: Regex matching IP addresses that are permitted to send data. Defaults to `.*` (that is, all IPs).

Max buffer size (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`.

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.

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 only for Cribl Stream deployments that are in Distributed mode.

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 **Group 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's data via the Routing table.

Select **QuickConnect** to send this Source's data 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 epoch time.
`HOSTNAME`	`host`	Yes	

RFC Name for Field	Cribl Name for Field	Guaranteed?	Notes
APP-NAME	appname	No	
PROCID	procid	No	
MSGID	msgid	No	
STRUCTURED-DATA	structuredData	No	
MSG	message	Yes	

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 later, `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
```

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
```

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 earlier 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
```

For the permanent settings change, add the following lines to `/etc/sysctl.conf`:

```
net.core.rmem_max=26214400
net.core.rmem_default=26214400
```

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 useful Advanced Troubleshooting short courses and Troubleshooting Criblets.

10.1.18. TCP JSON Source

Cribl Edge can receive newline-delimited JSON data over TCP.

Type: **Push** | TLS Support: **YES** | Event Breaker Support: **No**

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, unauthenticated 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)

Enabled defaults to `No`. When toggled to `Yes`:

Certificate name: 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. Defaults to `No`. When toggled to `Yes`:

- **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). Defaults to `Yes`.
- **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 this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

On Cribl-managed Cloud Workers (with certain plan/license tiers), 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 no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

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 Cribl Edge's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Edge's data processing engine.

Max buffer size: 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 per connection, not just per Worker Process – and this can dramatically expand memory usage.)

Commit frequency: The number of events to send downstream before committing that Cribl Stream has read them. Defaults to `42`.

Max file size: 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 Edge applies the default `1 MB`.

Max queue size: 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 **Max PQ size per Worker Process** in 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 Edge will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

In Cribl Edge 4.1 and later, Source-side PQ's default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Always On versus Smart Mode.

You can optimize Workers' startup connections and CPU load at **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 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 to process data from this input before the data is sent through the Routes.

Advanced Settings

Enable proxy protocol: Toggle to `Yes` if the connection is proxied by a device that supports proxy protocol v1 or v2.

Enable load balancing (Cribl Stream only): Toggle to `Yes` 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 only in Cribl Stream Distributed deployments.

IP allowlist regex: Regex matching IP addresses that are allowed to establish a connection. Defaults to `.*` (such as, all IPs).

Max active connections: 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.

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.

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":"v
{"host":"myOtherHost", "source":"myOtherSource", "_raw": "{\"message\":\"Something i
```

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 `<Your-Org-ID>` 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.<Your-Org-ID>.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 `<group-name>.main` when sending to other Fleets.

Port: `10070`

TLS Settings (Client Side)

Enabled: `Yes`

Validate server certs: `Yes`

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.

10.1.19. 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**

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 to `Yes` 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" }}`.
    - **Shared secret (authToken)**: Shared secret to be provided by any client (in `authToken` header field). Click **Generate** to create a new secret. If empty, unauthenticated 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)

Enabled defaults to `No`. When toggled to `Yes`:

Certificate name: 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. Defaults to `No`. When toggled to `Yes`:

- **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). Defaults to `Yes`.
- **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 this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

On Cribl-managed Cloud Workers (with certain plan/license tiers), 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 no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

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 Cribl Edge's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Edge's data processing engine.

Max buffer size: 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 per connection, not just per Worker Process – and this can dramatically expand memory usage.)

Commit frequency: The number of events to send downstream before committing that Stream has read them. Defaults to `42`.

Max file size: 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 Edge applies the default `1 MB`.

Max queue size: 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 **Max PQ size per Worker Process** in 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 Edge will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

In Cribl Edge 4.1 and later, Source-side PQ's default **Mode** is `Always On`, to best ensure events' delivery. For details on optimizing this selection, see Always On versus Smart Mode.

> You can optimize Workers' startup connections and CPU load at **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: Defaults to `No`. When toggled to `Yes`:

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), maxiumum `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 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 to process data from this input before the data is sent through the Routes.

Advanced Settings

Enable Proxy Protocol: Toggle to `Yes` 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).

Max active connections: 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: 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
```

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.<Your-Org-ID>.cribl.cloud 10060
```

With a Cribl.Cloud Enterprise plan, generalize the above URL's `default.main` substring to `<group-name>.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.

10.1.20. UDP (Raw) Source

Cribl Edge supports receiving raw, unparsed data via UDP.

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 this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

On Cribl-managed Cloud Workers (with certain plan/license tiers), 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 no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

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 Cribl Edge's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Edge's data processing engine.

Max buffer size: 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 per connection, not just per Worker Process – and this can dramatically expand memory usage.)

If you do not see the **Max buffer size** setting, it is because you enabled the Persistent Queue. The Persistent Queue provides its own configurable internal buffer, and **Max buffer size** is not applicable in that case.

Commit frequency: The number of events to send downstream before committing that Stream has read them. Defaults to `42`.

Max file size: 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, etc. If not specified, Cribl Edge applies the default `1 MB`.

Max queue size: 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`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in 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 Edge will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

In Cribl Edge 4.1 and later, Source-side PQ's default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Always On versus Smart Mode.

You can optimize Workers' startup connections and CPU load at **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 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 to process data from this input before the data is sent through the Routes.

Advanced Settings

Single msg per UDP: Toggle to `Yes` if each UDP message should be treated as an independent event. Leave set to the default `No` if the message should be broken on newlines to create multiple events.

Ingest raw bytes: Toggle to `Yes` 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).

Max buffer size (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 set to `Yes`), or of the given line that was split out of the message.
- `_time`: The UNIX timestamp (in 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 set to `Yes`, this field will be an array containing the bytes of the UDP message.

UDP Tuning

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 may be too small.

You can check the current buffer size with:

```
$ sysctl net.core.rmem_max
```

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
```

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 (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
```

For the permanent settings change, add the following lines to `/etc/sysctl.conf`:

```
net.core.rmem_max=26214400
net.core.rmem_default=26214400
```

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.

10.2. Destinations

Destinations are the endpoints where Cribl Edge delivers your data after processing and transformation. Cribl Edge can send data to many different Destinations, including Splunk, Kafka, Kinesis, InfluxDB, Snowflake, Databricks, TCP JSON, and 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.

Flow of data from Sources into Cribl Edge and then on to Destinations.

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 S3	HTTPS only	Non-Streaming		
Data Lakes > Amazon Security Lake	HTTP/S	Non-Streaming		
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		
Confluent Cloud	TCP	Streaming		
CrowdStrike Falcon LogScale	HTTPS only	Streaming		
CrowdStrike Falcon Next-Gen SIEM	HTTPS only	Streaming		
Datadog	HTTPS only	Streaming		
Dynatrace HTTP	HTTP/S	Streaming		
Dynatrace OTLP	HTTP/S	Streaming		

Destination	Protocol	Streaming	Filesystem-Based	Load-Balanced
Elastic Cloud	HTTPS only	Streaming		
Elasticsearch	HTTP/S	Streaming		
Exabeam	HTTP/S	Non-Streaming		
Filesystem/NFS		Non-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		
MinIO	HTTP/S	Non-Streaming		
NetFlow	UDP	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 DataSet	HTTPS only	Streaming		
ServiceNow Cloud	gRPC or HTTP/S	Streaming		

Observability Destination	Protocol	Streaming	Filesystem-Based	Load-Balanced
SignalFx	HTTPS only	Streaming		
SNMP Trap	UDP	Streaming		
Splunk HEC	HTTP/S	Streaming		
Splunk Load Balanced	TCP	Streaming		
Splunk Single Instance	TCP	Streaming		
StatsD Extended	TCP or UDP	Streaming		
StatsD	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.

- **SpaceOut**: This experimental Destination is undocumented. Be careful!

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.

10.2.1. Managing 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

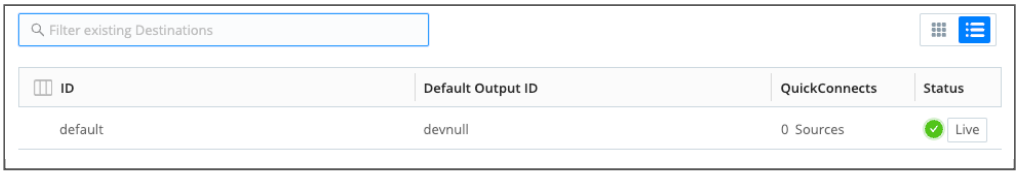
To edit any Destination's definition in a JSON text editor, select **Manage as JSON** at the bottom of the Destination modal. You can directly edit multiple values, and you can use the **Import** and **Export** buttons to

copy and modify existing Destination configurations as `.json` files.

When JSON configuration contains sensitive information, it is redacted during export.

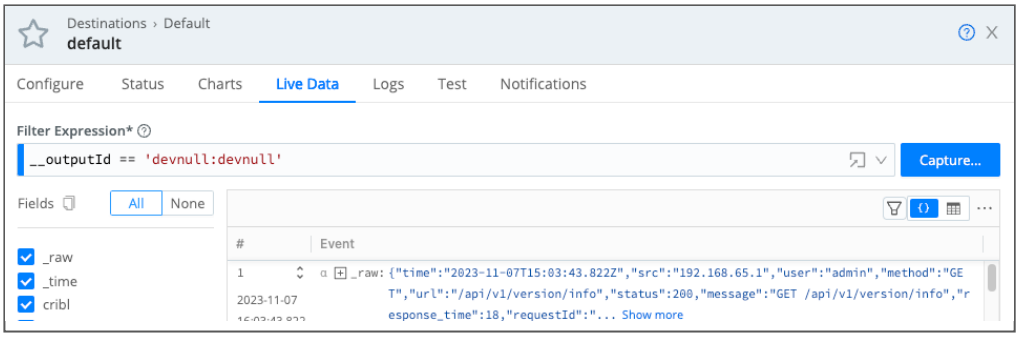
Capture Outgoing Data

To capture data from a single enabled Destination, you can bypass the 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.

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 Charts Tab

The **Charts** tab presents a visualization of the recent activity on the Destination. The following data is available:

- Events in
- Thruput in (events per second)
- Bytes in
- Thruput in (bytes per second)
- Blocked status

This data (in contrast with the status tab) is read almost instantly and does not reset when restarting an Edge Node.

10.2.2. Internal

10.2.2.1. Cribl HTTP Destination

The Cribl HTTP Destination sends data to a Cribl HTTP Source connected to the same Leader. Relaying data between a Cribl HTTP Destination and Cribl HTTP Source pair prevents double-billing: You'll be charged no additional charges or credits after initial ingest to the first Fleet.

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. All data subsequently relayed to other Workers via a Cribl HTTP Destination/Source pair is not charged.

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 Cribl HTTP Destination must be on a Node that is connected to the same Leader as the Cribl HTTP Source(s).
- 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 later can receive data only from Nodes running v.3.5.4 and later. Nodes running the Cribl HTTP Destination on Cribl 3.5.3 and earlier can receive data only from Nodes running v.3.5.3 and earlier.

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**: Set to `No` by default. When toggled to `Yes`, see Load Balancing Settings below. With the default `No` setting, if you notice that Cribl Edge is not sending data to all possible IP addresses, enable **Advanced Settings** > **Round-robin DNS**.
    - **Cribl endpoint**: URL of a Cribl Worker to send events to, for example, `http://localhost:10200`.

The **Cribl endpoint** field appears only when **Load balancing** is toggled to `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`.
    - **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 to `Yes`. 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 `No`. When toggled to `Yes`:

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 `Yes`. 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 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.

Persistent Queue Settings

This tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**.

> On Cribl-managed Cloud Workers (with certain plan tiers), 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 only when the Destination is in an error state.
Backpressure	After the Destination has been in a backpressure state for a specified amount of time, Cribl Stream/Edge queues and stores data to a disk until the backpressure event resolves.
Always on	Cribl Stream/Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

Max file size: 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`.

Max queue size: 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 **Max PQ size per Worker Process** on the Group Settings/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.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream/Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, switch this to `No`. When `No` is enabled, this option appears:

- **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 **Group Settings/Fleet Settings** > 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 **Output ID** has been defined.)

To learn more about persistent queue behavior and how to use it with your system, see:

- Persistent Queues
- Planning for Persistent Queues
- Configuring Persistent Queues

Processing Settings

Post-Processing

Pipeline: Pipeline 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: 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).

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 `Yes`. 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 to `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Stream to cycle through them in the order returned. 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`.

Max body size (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 **Max body size** 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.

10.2.2.2. Cribl TCP Destination

The Cribl TCP Destination sends data to a Cribl TCP Source connected to the same Leader. Relaying data between a Cribl TCP Destination and Cribl TCP Source pair prevents double-billing: You'll be charged no additional charges or credits after initial ingest to the first Fleet.

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 on another Worker, you're billed for ingress only once – when Cribl first receives the data. All data subsequently relayed to other Workers via a Cribl TCP Destination/Source pair is not charged.

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:

- The Cribl TCP Destination must be on a Node that is connected to the same Leader as the Cribl TCP Source.
- 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 later can send data only to Nodes running v.3.5.4 and later. Nodes running the Cribl TCP Source on Cribl 3.5.3 and earlier can send data only to Nodes running v.3.5.3 and earlier.

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 to `Yes`, see Load Balancing Settings below.

      The following two fields appear **only** with **Load balancing**'s default `No` setting, 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 **Max connections**, the throttling rate is applied per connection. For instance, if **Max connections** 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 to `Yes`. 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.

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

This tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**.

> On Cribl-managed Cloud Workers (with certain plan tiers), 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 only when the Destination is in an error state.
Backpressure	After the Destination has been in a backpressure state for a specified amount of time, Cribl Stream/Edge queues and stores data to a disk until the backpressure event resolves.
Always on	Cribl Stream/Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

Max file size: 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`.

Max queue size: 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 **Max PQ size per Worker Process** on the Group Settings/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.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream/Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, switch this to `No`. When `No` is enabled, this option appears:

- **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 **Group Settings/Fleet Settings** > 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 **Output ID** has been defined.)

To learn more about persistent queue behavior and how to use it with your system, see:

- Persistent Queues
- Planning for Persistent Queues
- Configuring Persistent Queues

TLS Settings (Client Side)

Use TLS defaults to `No`. When toggled to `Yes`:

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 `No`.

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 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.

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 to `Yes` makes the payloads of any (future) failed requests available for inspection. See Inspecting Payloads 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.

Inspecting Payloads to Troubleshoot Closed Connections

When a downstream receiver closes connections from this Destination (or just stops responding), inspecting the payloads of the resulting failed requests 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 payloads of failed requests. 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 payloads from failed requests. At the command line, navigate to that directory and inspect any payloads that you think might be relevant.

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`.

10.2.2.3. 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.

10.2.2.4. Cribl Stream Destination (Deprecated)

This Destination was deprecated as of Cribl Edge 3.5, and has been removed as of v.4.0. Please instead use the Cribl TCP or the Cribl HTTP Destination to enable Edge Nodes to send data to peer Nodes.

The Cribl Stream Destination enables Edge Nodes, and/or Cribl Stream instances, to send data to one or multiple Cribl Stream instances.

Type: Streaming | TLS Support: Configurable | PQ Support: Yes

Use New Destinations Instead

The replacement Destinations, Cribl TCP and Cribl HTTP, don't rely on IP filtering, for the following reasons:

- Load balancers and/or proxies between the Cribl Destination and Cribl Source can change the IP address, resulting in a bad match and rejected ingest.
- A Lookup table of all IP addresses needed to be sent to each Worker Node/Edge Node from the Leader, which is not scalable.
- The Lookup table of IP addresses required constant communication between the Worker Node/Edge Nodes and the Leader, making this fragile and placing an arbitrary reliance on the Leader that shouldn't be there.

Configuring a Cribl Stream Destination

In the **QuickConnect** UI: Click **Add Destination** beside **Destinations**. From the resulting drawer's tiles, select **Cribl Stream**. Next, click **Add Destination** to open a **New Cribl Stream** drawer.

Or, in the **Data Routes** UI: From the top nav of a Cribl Stream instance or Group, select **Data** > **Destinations**. From the top nav of a Cribl Edge instance or Fleet, select **More** > **Destinations**.

From the resulting page's tiles or the **Destinations** left nav, select **Cribl Stream**. Next, click **New Destination** to open a **Cribl Stream** > **New Destination** modal that provides the following options and fields.

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**.

Address: Hostname of the receiver.

Port: Port number to connect to on the host.

Authentication Settings

Use the **Authentication method** buttons to select one of these options:

- **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.

Optional Settings

Compression: Codec to use to compress the data before sending. Defaults to `None`.

Throttling: Throttle rate, in bytes per second. Defaults to `0`, meaning no throttling. Multiple-byte units such as KB, MB, GB etc. 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.

Persistent Queue Settings

This tab is displayed when the **Backpressure behavior** is set to **Persistent Queue**.

On Cribl-managed Cloud Workers (with certain plan/license tiers), this tab exposes only the destructive **Clear Persistent Queue** button (described below in 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 Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Max file size: 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`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Destination will stop queueing data and apply the **Queue-full** behavior. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Fleet Settings.

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.

Queue-full behavior: Whether to block or drop events when the queue is exerting backpressure (because disk is low or at full capacity). **Block** is the same behavior as non-PQ blocking, corresponding to the **Block** option on the **Backpressure behavior** drop-down. **Drop new data** throws away incoming data, while leaving the contents of the PQ unchanged.

Clear Persistent Queue: Click this "panic" button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear - because this will free up disk space by permanently deleting the queued data, without delivering it to downstream receivers. (Appears only after **Output ID** has been defined.)

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, toggle this off. Doing so will expose this additional control:

- **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's drain rate can boost the throughput of new/active connections, by reserving more resources for them. You can further optimize Workers' startup connections and CPU load at **Fleet Settings** > Worker Processes.

TLS Settings (Client Side)

Enabled defaults to `No`. When toggled to `Yes`:

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 `No`.

Server name (SNI): Server name for the SNI (Server Name Indication) TLS extension. This must be a host name, not an IP address.

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.

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.

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 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_wp` – Cribl Edge Worker Process that processed the event.
- `cribl_input` – Cribl Edge Source that processed the event.

- `cribl_output` – Cribl Edge Destination that processed the event.

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.

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.

10.2.2.5. 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.

10.2.2.6. 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.

10.2.2.7. Output Router Destination

Output Routers are meta-destinations that allow for output selection based on rules. Rules are evaluated in order, top-down, with the first match being the winner.

Type: Internal | TLS Support: N/A | PQ Support: N/A

Configure Cribl Edge to Send to an Output Router

In Cribl Edge, configure Output Router:

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. For details, see Rules below.
3. Next, you can configure the following **Optional Settings**:
    - **System fields**: A list of fields to automatically add to events that use this output. For details, see System Fields 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 Advanced settings outlined in the sections below.
5. Select **Save**, then **Commit & Deploy**.

Rules

Configure the following event routing rules settings.

- **Filter expression**: JavaScript expression to select events to send to output.
- **Output**: Output to send matching events to.
- **Description**: Optionally, enter a description of this rule's purpose.
- **Final**: Toggle to `No` if you want the event to be checked against other rules lower in the stack.

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.

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.

Limitations/Options

- An Output Router cannot reference another. This is by design, to avoid circular references.
- Also to avoid circular references, an Output Router cannot reference a Default Destination that points back to Output Router.
- **Events that do not match any of the rules are dropped**. Use a catch-all rule to change this behavior.
- No post-processing (conditioning) can be attached to the Output Router Destination itself. But you are free to use pre-processing Pipelines on the Source tier, and post-processing Pipelines on any or all Destinations that the Output Router references.
- Data can be cloned by toggling the `Final` flag to `No`. (The default is `Yes`, i.e., no cloning.)

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
`method=='POST' || method=='GET'`	**Los Angeles**	Yes
`true`	**New York**	Yes

10.2.3. Amazon

10.2.3.1. 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

This tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**.

> On Cribl-managed Cloud Workers (with certain plan tiers), 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 only when the Destination is in an error state.
Backpressure	After the Destination has been in a backpressure state for a specified amount of time, Cribl Stream/Edge queues and stores data to a disk until the backpressure event resolves.
Always on	Cribl Stream/Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

Max file size: 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`.

Max queue size: 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 **Max PQ size per Worker Process** on the Group Settings/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.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream/Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, switch this to `No`. When `No` is enabled, this option appears:

- **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 **Group Settings/Fleet Settings** > 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 **Output ID** has been defined.)

To learn more about persistent queue behavior and how to use it with your system, see:

- Persistent Queues
- Planning for Persistent Queues
- Configuring Persistent Queues

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. **IAM Roles for Amazon EC2**: Loaded from AWS Identity and Access Management (IAM) roles attached to an EC2 instance.
2. **Shared Credentials File**: Loaded from the shared credentials file (`~/.aws/credentials`).
3. **Environment Variables**: Loaded from environment variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`.
4. **JSON File on Disk**: Loaded from a JSON file on disk.
5. **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 Stream, 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. Setting an invalid region results in a fallback to the global STS endpoint.

Enable for CloudWatch Logs: Toggle to `Yes` 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 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.

Max queue size: Maximum number of queued batches before blocking. Defaults to `5`.

Max record size (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. The default setting (`Yes`) 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 `Yes`.

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.

10.2.3.2. 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:

- `kinesis:DescribeStream` - Always required.
- `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

This tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**.

> On Cribl-managed Cloud Workers (with certain plan tiers), 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 only when the Destination is in an error state.
Backpressure	After the Destination has been in a backpressure state for a specified amount of time, Cribl Stream/Edge queues and stores data to a disk until the backpressure event resolves.
Always on	Cribl Stream/Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

Max file size: 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`.

Max queue size: 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 **Max PQ size per Worker Process** on the Group Settings/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.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream/Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, switch this to `No`. When `No` is enabled, this option appears:

- **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 **Group Settings/Fleet Settings** > 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 **Output ID** has been defined.)

To learn more about persistent queue behavior and how to use it with your system, see:

- Persistent Queues
- Planning for Persistent Queues
- Configuring Persistent Queues

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. **IAM Roles for Amazon EC2**: Loaded from AWS Identity and Access Management (IAM) roles attached to an EC2 instance.
2. **Shared Credentials File**: Loaded from the shared credentials file (`~/.aws/credentials`).
3. **Environment Variables**: Loaded from environment variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`.
4. **JSON File on Disk**: Loaded from a JSON file on disk.
5. **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 Stream, 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. Setting an invalid region results in a fallback to the global STS endpoint.

Enable for Kinesis Streams: Toggle to `Yes` 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 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`.

Max record size (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. The default setting (`Yes`) 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 `Yes`.

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:

- **Max records per flush**: Maximum number of records to send in a single request to the Amazon Kinesis Data Streams Service PutRecords API 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.

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 re
{"_raw":"07-03-2018 18:33:51.136 -0700 INFO  TcpOutputProc - Connection to 127.0.0.1
...
```

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.

10.2.3.3. Amazon S3 Compatible Stores Destination

S3 is a non-streaming Destination type. Cribl Edge does **not** have to run on AWS in order 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 services like Databricks and 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.
   - **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**.
   - **Compress**: 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. **IAM Roles for Amazon EC2**: Loaded from AWS Identity and Access Management (IAM) roles attached to an EC2 instance.
2. **Shared Credentials File**: Loaded from the shared credentials file (`~/.aws/credentials`).
3. **Environment Variables**: Loaded from environment variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`.
4. **JSON File on Disk**: Loaded from a JSON file on disk.
5. **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 Stream, 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. Setting an invalid region results in a fallback to the global STS endpoint.

Enable for S3: Toggle to `Yes` 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 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 Working with Parquet for pointers on how to avoid problems such as data mismatches.
- In Cribl Edge 4.3 and later, 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 to `Yes` 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

Max file size (MB): Maximum uncompressed output file size. Files of this size will be closed and moved to final output location. Defaults to `32`.

Max file open time (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`.

Max file idle time (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`.

Max open files: 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.

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.

Max concurrent file parts: 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 set to `Yes` (the 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 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 staging dirs** is enabled. Defaults to `300` seconds (every 5 minutes). Minimum configurable interval is `10` seconds; maximum is `86400` seconds (every 24 hours).

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. The default setting (`Yes`) 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 `Yes`.

Verify if bucket exists: Toggle this to `No` 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 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.

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
```

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.

10.2.3.4. 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 `Yes`.
    - **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

This tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**.

> On Cribl-managed Cloud Workers (with certain plan tiers), 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 only when the Destination is in an error state.
Backpressure	After the Destination has been in a backpressure state for a specified amount of time, Cribl Stream/Edge queues and stores data to a disk until the backpressure event resolves.
Always on	Cribl Stream/Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

Max file size: 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`.

Max queue size: 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 **Max PQ size per Worker Process** on the Group Settings/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.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream/Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, switch this to `No`. When `No` is enabled, this option appears:

- **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 **Group Settings/Fleet Settings** > 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 **Output ID** has been defined.)

To learn more about persistent queue behavior and how to use it with your system, see:

- Persistent Queues
- Planning for Persistent Queues
- Configuring Persistent Queues

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. **IAM Roles for Amazon EC2**: Loaded from AWS Identity and Access Management (IAM) roles attached to an EC2 instance.
2. **Shared Credentials File**: Loaded from the shared credentials file (`~/.aws/credentials`).
3. **Environment Variables**: Loaded from environment variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`.
4. **JSON File on Disk**: Loaded from a JSON file on disk.
5. **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 Stream, 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. Setting an invalid region results in a fallback to the global STS endpoint.

Enable for SQS: Toggle to `Yes` 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 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`.

Max queue size: Maximum number of queued batches before blocking. Defaults to `100`.

Max record size (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`.

Max concurrent requests: The maximum number of in-progress API requests before backpressure is applied. Defaults to `10`.

Reuse connections: Whether to reuse connections between requests. The default setting (`Yes`) 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 `Yes`.

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.

10.2.3.5. 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

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`. 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, 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 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

Row group size: Ideal memory size for row group segments. For example, 128 MB or 1 GB. Affects memory use when writing. Imposes a target, not a strict limit; the final size of a row group may be larger or smaller. Defaults to 16 MB.

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.

Parquet version: Which version of Parquet your schemas are in determines which data types are supported and how they are represented. Defaults to version 2.6.

Data page version: Serialization format of data pages. Note that not all reader implementations support Data page V2.

Cribl Edge Workers support Parquet only when running on Linux, not on Windows.

Advanced Settings

Max file size (MB): Maximum uncompressed output file size. Files of this size will be closed and moved to final output location. Defaults to `32`.

Max file open time (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`.

Max file idle time (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`.

Max open files: 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.

Max concurrent file parts: 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 set to `Yes` (the 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 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. Defaults to `300` seconds (every 5 minutes). Minimum configurable interval is `10` seconds; maximum is `86400` seconds (every 24 hours).

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. The default setting (`Yes`) 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 `Yes`.

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 here.

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.

10.2.4. Azure

10.2.4.1. 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.

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**: Enter the container name. (A container organizes a set of blobs, similar to a directory in a file system.)

      Container names can include only lowercase letters, numbers, and/or hyphens (-). This restriction is imposed by Azure.
    - **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.

      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 to `Yes` to create the configured container in Azure Blob Storage if one does not already exist.
   - **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**.
   - **Compress**: 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**: 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=<yo
```

A fictitious example, using Microsoft's recommended HTTPS option, is:

```
DefaultEndpointsProtocol=https;AccountName=storagesample;AccountKey=12345678...32;En
```

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.

Processing Settings

Post-Processing

Pipeline: Pipeline 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 Working with Parquet 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 to `Yes` 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

Max file size (MB): Maximum uncompressed output file size. Files reaching this size will be closed and moved to the final output location. Defaults to `32`.

Max file open time (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`.

Max file idle time (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`.

Max open files: 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 `Max file size (MB)` or the `Max file open time (sec)` conditions are met.

Max concurrent file parts: 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: When set to `Yes` (the 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 staging dirs: 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).

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`.

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`

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 and OIDC (On-Prem).
- SSO with Microsoft Entra ID and SAML (Cribl.Cloud).
- SSO with Microsoft Entra ID and SAML (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.

10.2.4.2. 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
```

Here's an example query:

```
.show table SyslogTable ingestion json mapping "SyslogMapping"
```

The Kusto query that originally created this mapping would look like this:

```
.create table SyslogTable ingestion json mapping "SyslogMapping"
```

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 `Yes`. Disable 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.
- **Compress**: 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

This tab is displayed when **General Settings** > **Ingestion mode** is set to **Streaming** mode and **Backpressure behavior** is set to **Persistent Queue**.

> On Cribl-managed Cloud Edge Nodes (with certain plan/license tiers), this tab should expose only the destructive **Clear Persistent Queue** button (described below in this section).
>
> Due to Known Issue CRIBL-21335, the on-prem Persistent Queue settings are still displayed in Cribl.Cloud – but you should ignore all controls other than the **Clear Persistent Queue** button.
>
> The other controls are not needed because Cribl.Cloud automatically allocates a maximum queue size of 1 GB disk space 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 Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Max file size: 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`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Destination will stop queueing data and apply the **Queue-full** behavior. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Fleet Settings.

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.

Queue-full behavior: Whether to block or drop events when the queue is exerting backpressure (because disk is low or at full capacity). **Block** is the same behavior as non-PQ blocking, corresponding to the `Block` option on the **Backpressure behavior** drop-down. **Drop new data** throws away incoming data, while leaving the contents of the PQ unchanged.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, toggle this off. Doing so will expose this additional control:

- **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's drain rate can boost the throughput of new/active connections, by reserving more resources for them. You can further optimize Workers' startup connections and CPU load at **Fleet Settings** > Worker Processes.

Clear Persistent Queue: Click this "panic" button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear - because this will free up disk space by permanently deleting the queued data, without delivering it to downstream receivers. (Appears only after **Output ID** has been defined.)

Processing Settings

Post-Processing

Pipeline: Pipeline 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 Working with Parquet 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 to `Yes` 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.

Max file size (MB): Maximum uncompressed output file size. Files reaching this size will be closed and moved to the storage container. Defaults to `32`.

Max file open time (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).

Max file idle time (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`.

Max open files: 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`.

Max concurrent file parts: 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 dirs: 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).

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`.

Max body size (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 **Max body size** 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 settings could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Validate server certs: When toggled on (the 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 to `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Edge to cycle through them in the order returned.

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.

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 and OIDC (On-Prem).
- SSO with Microsoft Entra ID and SAML (Cribl.Cloud).
- SSO with Microsoft Entra ID and SAML (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.

10.2.4.3. 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`.
- **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 `Yes`.

Validate server certs: Defaults to `Yes`.

Authentication

Authentication parameters to use when connecting to brokers. Using TLS is highly recommended.

Enabled: With the default `Yes` setting, 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

This tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**.

On Cribl-managed Cloud Workers (with certain plan tiers), 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 only when the Destination is in an error state.
Backpressure	After the Destination has been in a backpressure state for a specified amount of time, Cribl Stream/Edge queues and stores data to a disk until the backpressure event resolves.
Always on	Cribl Stream/Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

Max file size: 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`.

Max queue size: 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 **Max PQ size per Worker Process** on the Group Settings/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.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream/Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, switch this to `No`. When `No` is enabled, this option appears:

- **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 **Group Settings/Fleet Settings** > 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 **Output ID** has been defined.)

To learn more about persistent queue behavior and how to use it with your system, see:

- Persistent Queues
- Planning for Persistent Queues
- Configuring Persistent Queues

Processing Settings

Post-Processing

Pipeline: Pipeline 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.

Max retries: 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

Max record size (KB, uncompressed): Maximum size (KB) of each record batch before compression. Setting should be < `message.max.bytes` settings in Kafka brokers. Defaults to `768`.

Max events per batch: 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 epoch time (either seconds or milliseconds since the UNIX epoch – 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`

Resources

For examples of configuring Cribl Edge to interoperate with Azure services, see these guides:

- Preparing the Azure Workspace.
- SSO with Microsoft Entra ID and OIDC (On-Prem).
- SSO with Microsoft Entra ID and SAML (Cribl.Cloud).
- SSO with Microsoft Entra ID and SAML (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, 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.

10.2.4.4. Azure Monitor Logs Destination

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

This tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**.

> On Cribl-managed Cloud Workers (with certain plan tiers), 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 only when the Destination is in an error state.
Backpressure	After the Destination has been in a backpressure state for a specified amount of time, Cribl Stream/Edge queues and stores data to a disk until the backpressure event resolves.
Always on	Cribl Stream/Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

Max file size: 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`.

Max queue size: 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 **Max PQ size per Worker Process** on the Group Settings/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.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream/Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, switch this to `No`. When `No` is enabled, this option appears:

- **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 **Group Settings/Fleet Settings** > 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 **Output ID** has been defined.)

To learn more about persistent queue behavior and how to use it with your system, see:

- Persistent Queues
- Planning for Persistent Queues
- Configuring Persistent Queues

Processing Settings

Post-Processing

Pipeline: Pipeline 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
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).

Advanced Settings

Validate server certs: Toggle to `Yes` 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 to `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Edge to cycle through them in the order returned.

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`.

Max body size (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 **Max body size** 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 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.

Resources

For examples of configuring Cribl Edge to interoperate with Azure services, see these guides:

- Preparing the Azure Workspace.
- SSO with Microsoft Entra ID and OIDC (On-Prem).
- SSO with Microsoft Entra ID and SAML (Cribl.Cloud).
- SSO with Microsoft Entra ID and SAML (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.

10.2.4.5. 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.

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. In this case, Cribl recommends you use a single URL to ingest all the data, with an Output Router to route each event type to the correct Microsoft Sentinel Destination.

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

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: Data collection endpoint (DCE) in the format `https://<endpoint-name>.<identifier>.<region>.ingest.monitor.azure.com`.

Data collection rule ID: Immutable ID for the data collection rule (DCR).

Stream name: Name of the Sentinel table in which to store events.

Optional Settings

Authentication Settings

Login URL: Endpoint for the OAuth API call. This URL varies depending on your Azure environment:

- 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.

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`

Persistent Queue Settings

This tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**.

On Cribl-managed Cloud Workers (with certain plan tiers), 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 only when the Destination is in an error state.
Backpressure	After the Destination has been in a backpressure state for a specified amount of time, Cribl Stream/Edge queues and stores data to a disk until the backpressure event resolves.
Always on	Cribl Stream/Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

Max file size: 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`.

Max queue size: 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 **Max PQ size per Worker Process** on the Group Settings/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.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream/Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, switch this to `No`. When `No` is enabled, this option appears:

- **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 **Group Settings/Fleet Settings** > 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 **Output ID** has been defined.)

To learn more about persistent queue behavior and how to use it with your system, see:

- Persistent Queues
- Planning for Persistent Queues
- Configuring Persistent Queues

Processing Settings

Post-Processing

Pipeline: Pipeline 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
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).

Advanced Settings

Validate server certs: Reject certificates that are not authorized by a trusted CA (for example, the system's CA). Toggle this to `No` if you want Cribl Edge to accept such certificates. Defaults to `Yes`.

Round-robin DNS: Toggle to `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Edge to cycle through them in the order returned.

Compress: 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`.

Max body size (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 **Max body size** 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 **Max Body Size** 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: 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.

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.

10.2.5. Elastic

10.2.5.1. 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.

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**: When enabled (default), lets you specify multiple bulk API URLs and load weights. With the `No` setting, if 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** to `Yes` 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 set to `Yes`. It 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.
    - **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

This tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**.

On Cribl-managed Cloud Workers (with certain plan tiers), 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 only when the Destination is in an error state.
Backpressure	After the Destination has been in a backpressure state for a specified amount of time, Cribl Stream/Edge queues and stores data to a disk until the backpressure event resolves.
Always on	Cribl Stream/Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

Max file size: 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`.

Max queue size: 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 **Max PQ size per Worker Process** on the Group Settings/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.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream/Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, switch this to `No`. When `No` is enabled, this option appears:

- **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 **Group Settings/Fleet Settings** > 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 **Output ID** has been defined.)

To learn more about persistent queue behavior and how to use it with your system, see:

- Persistent Queues
- Planning for Persistent Queues
- Configuring Persistent Queues

Processing Settings

Post-Processing

Pipeline: Pipeline 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
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).

Advanced Settings

Validate server certs: Reject certificates that are **not** authorized by a trusted CA (for example, the system's CA). Defaults to `Yes`.

Round-robin DNS: Toggle to `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Edge to cycle through them in the order returned. (This option is visible only when the **General Settings** > **Load balancing** option is set to `No`.)

Compress: Compresses the payload body before sending. Defaults to `Yes` (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`.

Max body size (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 **Max body size** 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 (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 the **General Settings** > **Load balancing** option is set to `Yes`.

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 this setting to `No` 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

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.
- 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.

10.2.5.2. Elastic Cloud Destination

Cribl Edge can send events to Elastic Cloud.

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

This tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**.

> On Cribl-managed Cloud Workers (with certain plan tiers), 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 only when the Destination is in an error state.
Backpressure	After the Destination has been in a backpressure state for a specified amount of time, Cribl Stream/Edge queues and stores data to a disk until the backpressure event resolves.
Always on	Cribl Stream/Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

Max file size: 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`.

Max queue size: 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 **Max PQ size per Worker Process** on the Group Settings/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.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream/Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, switch this to `No`. When `No` is enabled, this option appears:

- **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 **Group Settings/Fleet Settings** > 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 **Output ID** has been defined.)

To learn more about persistent queue behavior and how to use it with your system, see:

- Persistent Queues
- Planning for Persistent Queues
- Configuring Persistent Queues

Processing Settings

Post-Processing

Pipeline: Pipeline 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
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).

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`.

Max body size (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 **Max body size** 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 (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 this setting to `No` 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.

10.2.6. Google Cloud

10.2.6.1. 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.

Type: Streaming | TLS Support: Yes | PQ Support: Yes

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.
    - **Send events as**: From the drop-down, choose how you want the Destination to send data to SecOps. The options are: **Unstructured** and **UDM**. See Options for Sending Data to SecOps below to learn about the differences between working with `Unstructured` and already-structured `UDM` data.
    - **API version**: The version of the Google Security Operations Ingestion API you want the Destination to use when communicating with SecOps. Defaults to `V2`.

- **Default log type**: This setting is available only when **Send events as** is set to `Unstructured` (the default). From the drop-down, select a log type to assign to events that lack a `__logType` field. For any events that **do** have a `__logType` field, Cribl Edge uses the value in that field, not the default log type. See Working with Log Types below to learn how Cribl Edge populates the drop-down and how to add log types missing from it.
  When **API version** is set to `V2` (the default), one or both of the following additional controls appear in **General Settings**:
- **Namespace**: This setting is available only when **Send events as** is set to `Unstructured` (the default). Optionally, 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. To use this optional field, request the ID from your SecOps representative.
  When **API version** is set to `V1` and **Send events as** is set to `Unstructured`, both **Namespace** and **Customer ID** appear under **Optional Settings**.

3. Next, you can configure the following **Optional Settings**: When **General Settings** > **Send events as** is set to `Unstructured` (the default), the UI displays some or all of these settings, depending on how **API version** is set.
   - **Custom log types**: For each log type you want to define, click **Add type** and enter values for **Log type** and **Description** in the resulting table row. See Working with Log Types above for more details.
   - **Namespace**: Appears here when **API version** is set to `V1`. Described in **General Settings** above.
   - **Customer ID**: Appears here when **API version** is set to `V1`. Described in **General Settings** above.
   - **Custom labels**: You can (optionally) define labels for the Destination to add to events. These "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.
   - **Log text field**: Specify the event field that contains the log text to send. If you do not specify a log text field, Cribl Edge sends a JSON representation of the whole event.

     (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.

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**.

Working with Log Types

Google SecOps supports a dynamic list of log types that evolves over time. Cribl Edge periodically updates the **Default log type** drop-down with the latest list from Google. However, if you need to use a log type that is not yet included in the drop-down or an older log type that has been removed, you can define and manage these log types using the **Custom log types** table under **Optional Settings**.

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.

For Google's latest list of supported log types, see the API documentation or send a `GET` request to https://malachiteingestion-pa.googleapis.com/v1/logtypes.

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** drop-down 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

This tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**.

> On Cribl-managed Cloud Workers (with certain plan tiers), 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 only when the Destination is in an error state.
Backpressure	After the Destination has been in a backpressure state for a specified amount of time, Cribl Stream/Edge queues and stores data to a disk until the backpressure event resolves.
Always on	Cribl Stream/Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

Max file size: 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`.

Max queue size: 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 **Max PQ size per Worker Process** on the Group Settings/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.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream/Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, switch this to `No`. When `No` is enabled, this option appears:

- **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 **Group Settings/Fleet Settings** > 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 **Output ID** has been defined.)

To learn more about persistent queue behavior and how to use it with your system, see:

- Persistent Queues
- Planning for Persistent Queues
- Configuring Persistent Queues

Options for Sending Data to SecOps

The options for Sending Data to SecOps are:

- Sending `Unstructured` data that SecOps then parses into UDM (Unified Data Model).
- Sending data **already structured** as `UDM`, bypassing the SecOps parsers.

Sending unstructured data to SecOps is best for writing **well-known data types** to SecOps. SecOps runs the unstructured data through a series of parsers that transform it to satisfy the required UDM schema. Where this can break down is when you need to write less-usual data types, which can stump SecOps's parsers, leading to parsing errors and additional time spent troubleshooting.

Sending data already structured as UDM into SecOps bypasses the parsing gauntlet. When the customer is working with **less-usual or unique, proprietary data types** that could confound the parsers, this can be essential.

Overall, support for sending data already structured as UDM provides more control and flexibility to land more data in SecOps. To gain that flexibility, you must be prepared to "own" the data transformation into UDM before sending it to SecOps – this method "gives you the keys." You will need to use Pipelines and Functions to transform your initially unstructured data to UDM before routing it to this Destination.

Processing Settings

Post-Processing

Pipeline: In this section's **Pipeline** drop-down list, you can select a single existing Pipeline 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
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, `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, 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 `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 `Yes`.

Round-robin DNS: Enable to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Edge to cycle through them in the order returned.

Compress: Compresses the payload body before sending. Defaults to `Yes` (recommended).

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.

Max body size (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 **Max body size** 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 **Max Body Size** settings.

Max events per request: 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 **Max body size**.

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.

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.

10.2.6.2. 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.
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. Optionally, you can adjust the Authentication, Processing, and Advanced settings outlined in the sections below.
6. 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

This tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**.

> On Cribl-managed Cloud Workers (with certain plan tiers), 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 only when the Destination is in an error state.
Backpressure	After the Destination has been in a backpressure state for a specified amount of time, Cribl Stream/Edge queues and stores data to a disk until the backpressure event resolves.
Always on	Cribl Stream/Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

Max file size: 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`.

Max queue size: 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 **Max PQ size per Worker Process** on the Group Settings/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.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream/Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, switch this to `No`. When `No` is enabled, this option appears:

- **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 **Group Settings/Fleet Settings** > 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 **Output ID** has been defined.)

To learn more about persistent queue behavior and how to use it with your system, see:

- Persistent Queues
- Planning for Persistent Queues
- Configuring Persistent Queues

Processing Settings

Post-Processing

Pipeline: Pipeline 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.

Max body size (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 **Max body size** 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 **Max Body Size** settings.

Max events per request: 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.

10.2.6.3. 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 to `Yes` if you want Cribl Edge to create the topic on Pub/Sub if it does not exist.
- **Ordered delivery**: Toggle to `Yes` 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

This tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**.

> On Cribl-managed Cloud Workers (with certain plan tiers), 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 only when the Destination is in an error state.
Backpressure	After the Destination has been in a backpressure state for a specified amount of time, Cribl Stream/Edge queues and stores data to a disk until the backpressure event resolves.
Always on	Cribl Stream/Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

Max file size: 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`.

Max queue size: 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 **Max PQ size per Worker Process** on the Group Settings/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.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream/Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, switch this to `No`. When `No` is enabled, this option appears:

- **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 **Group Settings/Fleet Settings** > 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 **Output ID** has been defined.)

To learn more about persistent queue behavior and how to use it with your system, see:

- Persistent Queues
- Planning for Persistent Queues
- Configuring Persistent Queues

Processing Settings

Post-Processing

Pipeline: Pipeline 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.

Max queue size: Maximum number of queued batches before blocking. Defaults to `100`.

Max batch size (KB): Maximum size for each sent batch. Defaults to `256` KB.

Max concurrent requests: 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.

10.2.6.4. 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 Google service account or user can have the Storage Admin or Owner role. For narrower access, the Google account should have the following access to buckets:

- `roles/storage.objectCreator` role.
- `roles/storage.objectViewer` role.
- If the `Verify if bucket exists` advanced option is enabled, one bucket-level `storage.buckets.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**.
   - **Compress**: 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 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 Working with Parquet 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 to `Yes` 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

Max file size (MB): Maximum uncompressed output file size. Files of this size will be closed and moved to final output location. Defaults to `32`.

Max file open time (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`.

Max file idle time (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`.

Max open files: 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 `Max file size (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 `Yes`.

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 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 staging dirs** is enabled. Defaults to `300` seconds (every 5 minutes). Minimum configurable interval is `10` seconds; maximum is `86400` seconds (every 24 hours).

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: 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. The default setting (`Yes`) 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 `Yes`.

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.

10.2.7. Kafka

10.2.7.1. 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.
    - **Brokers**: List of Kafka brokers 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`, or `LZ4`.

      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

This tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**.

> On Cribl-managed Cloud Workers (with certain plan tiers), 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 only when the Destination is in an error state.
Backpressure	After the Destination has been in a backpressure state for a specified amount of time, Cribl Stream/Edge queues and stores data to a disk until the backpressure event resolves.
Always on	Cribl Stream/Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

Max file size: 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`.

Max queue size: 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 **Max PQ size per Worker Process** on the Group Settings/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.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream/Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, switch this to `No`. When `No` is enabled, this option appears:

- **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 **Group Settings/Fleet Settings** > 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 **Output ID** has been defined.)

To learn more about persistent queue behavior and how to use it with your system, see:

- Persistent Queues
- Planning for Persistent Queues
- Configuring Persistent Queues

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 **Max retries**. 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 Defaults to `No`. When toggled to `Yes`:

- **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 `No`.

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 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.

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: Defaults to `No`. When toggled to `Yes`:

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 Avro-encoded data with a schema stored in the Confluent Schema Registry.

Enabled: defaults to `No`. When toggled to `Yes`, 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 to `Yes` 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 `No`. When toggled to `Yes`, 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). Defaults to `No`.
- **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.

Processing Settings

Post-Processing

Pipeline: Pipeline 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.

Max retries: 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

Max record size (KB, uncompressed): Maximum size (KB) of each record batch before compression. Setting should be < `message.max.bytes settings` in Kafka brokers. Defaults to `768`.

Max events per batch: 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 Data Sources 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 entails working with hostnames for **brokers** and **bootstrap servers**.

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** > **Brokers** 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** > **Brokers** 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 epoch time (either seconds or milliseconds since the UNIX epoch – 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.

10.2.7.2. 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`, or `LZ4`. 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

This tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**.

> On Cribl-managed Cloud Workers (with certain plan tiers), 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 only when the Destination is in an error state.
Backpressure	After the Destination has been in a backpressure state for a specified amount of time, Cribl Stream/Edge queues and stores data to a disk until the backpressure event resolves.
Always on	Cribl Stream/Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

Max file size: 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`.

Max queue size: 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 **Max PQ size per Worker Process** on the Group Settings/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.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream/Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, switch this to `No`. When `No` is enabled, this option appears:

- **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 **Group Settings/Fleet Settings** > 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 **Output ID** has been defined.)

To learn more about persistent queue behavior and how to use it with your system, see:

- Persistent Queues
- Planning for Persistent Queues
- Configuring Persistent Queues

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 **Max retries**. 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 to `Yes` (the default):

Autofill?: This setting is experimental.

Validate server certs: Toggle to `Yes` 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 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.

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: Defaults to `No`. When toggled to `Yes`:

- **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 Avro-encoded data with a schema stored in the Confluent Schema Registry.

Enabled: Defaults to `No`. When toggled to `Yes`, 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 to `Yes` 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 `No`. When toggled to `Yes`, 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). Defaults to `No`.
- **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 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.

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.

Max retries: 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

Max record size (KB, uncompressed): Maximum size (KB) of each record batch before compression. Setting should be < `message.max.bytes settings` in Kafka brokers. Defaults to `768`.

Max events per batch: 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 Data Sources 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.

10.2.7.3. 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.
    - **Brokers**: List of Kafka brokers 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`, or `LZ4`. 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

This tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**.

> On Cribl-managed Cloud Workers (with certain plan tiers), 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 only when the Destination is in an error state.
Backpressure	After the Destination has been in a backpressure state for a specified amount of time, Cribl Stream/Edge queues and stores data to a disk until the backpressure event resolves.
Always on	Cribl Stream/Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

Max file size: 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`.

Max queue size: 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 **Max PQ size per Worker Process** on the Group Settings/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.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream/Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, switch this to `No`. When `No` is enabled, this option appears:

- **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 **Group Settings/Fleet Settings** > 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 **Output ID** has been defined.)

To learn more about persistent queue behavior and how to use it with your system, see:

- Persistent Queues
- Planning for Persistent Queues
- Configuring Persistent Queues

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 **Max retries**. 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 `Yes`.

Server name (SNI): Leave this field blank. See Connecting to Kafka below.

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.

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. **IAM Roles for Amazon EC2**: Loaded from AWS Identity and Access Management (IAM) roles attached to an EC2 instance.
2. **Shared Credentials File**: Loaded from the shared credentials file (`~/.aws/credentials`).
3. **Environment Variables**: Loaded from environment variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`.
4. **JSON File on Disk**: Loaded from a JSON file on disk.
5. **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 Stream, 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. Setting an invalid region 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 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.

Max retries: 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

Max record size (KB, uncompressed): Maximum size (KB) of each record batch before compression. Setting should be < `message.max.bytes settings` in Kafka brokers. Defaults to `768`.

Max events per batch: 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 Data Sources 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 entails working with hostnames for **brokers** and **bootstrap servers**.

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** > **Brokers** 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** > **Brokers** 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 epoch time (either seconds or milliseconds since the UNIX epoch – 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.

10.2.8. Metrics

10.2.8.1. 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

This tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**.

> On Cribl-managed Cloud Workers (with certain plan tiers), 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 only when the Destination is in an error state.
Backpressure	After the Destination has been in a backpressure state for a specified amount of time, Cribl Stream/Edge queues and stores data to a disk until the backpressure event resolves.
Always on	Cribl Stream/Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

Max file size: 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`.

Max queue size: 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 **Max PQ size per Worker Process** on the Group Settings/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.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream/Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, switch this to `No`. When `No` is enabled, this option appears:

- **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 **Group Settings/Fleet Settings** > 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 **Output ID** has been defined.)

To learn more about persistent queue behavior and how to use it with your system, see:

- Persistent Queues
- Planning for Persistent Queues
- Configuring Persistent Queues

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 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

Max record size (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.

10.2.8.2. 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

This tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**.

On Cribl-managed Cloud Workers (with certain plan tiers), 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 only when the Destination is in an error state.
Backpressure	After the Destination has been in a backpressure state for a specified amount of time, Cribl Stream/Edge queues and stores data to a disk until the backpressure event resolves.
Always on	Cribl Stream/Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

Max file size: 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`.

Max queue size: 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 **Max PQ size per Worker Process** on the Group Settings/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.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream/Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, switch this to `No`. When `No` is enabled, this option appears:

- **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 **Group Settings/Fleet Settings** > 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 **Output ID** has been defined.)

To learn more about persistent queue behavior and how to use it with your system, see:

- Persistent Queues
- Planning for Persistent Queues
- Configuring Persistent Queues

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 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

Max record size (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.

10.2.8.3. 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

This tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**.

> On Cribl-managed Cloud Workers (with certain plan tiers), 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 only when the Destination is in an error state.
Backpressure	After the Destination has been in a backpressure state for a specified amount of time, Cribl Stream/Edge queues and stores data to a disk until the backpressure event resolves.
Always on	Cribl Stream/Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

Max file size: 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`.

Max queue size: 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 **Max PQ size per Worker Process** on the Group Settings/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.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream/Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, switch this to `No`. When `No` is enabled, this option appears:

- **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 **Group Settings/Fleet Settings** > 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 **Output ID** has been defined.)

To learn more about persistent queue behavior and how to use it with your system, see:

- Persistent Queues
- Planning for Persistent Queues
- Configuring Persistent Queues

Processing Settings

Post-Processing

Pipeline: Pipeline 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

Max record size (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.

10.2.9. New Relic Ingest

10.2.9.1. 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

This tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**.

> On Cribl-managed Cloud Workers (with certain plan tiers), 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 only when the Destination is in an error state.

Mode	Description
Backpressure	After the Destination has been in a backpressure state for a specified amount of time, Cribl Stream/Edge queues and stores data to a disk until the backpressure event resolves.
Always on	Cribl Stream/Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

Max file size: 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`.

Max queue size: 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 **Max PQ size per Worker Process** on the Group Settings/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.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream/Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, switch this to `No`. When `No` is enabled, this option appears:

- **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 **Group Settings/Fleet Settings** > 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 **Output ID** has been defined.)

To learn more about persistent queue behavior and how to use it with your system, see:

- Persistent Queues
- Planning for Persistent Queues
- Configuring Persistent Queues

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: 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).

Advanced Settings

Validate server certs: Toggle to `Yes` 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 to `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Edge to cycle through them in the order returned.

Compress: Defaults to `Yes`, 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`.

Max body size (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 **Max body size** value until downstream receivers reliably accept all events.

Max events per request: 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 **Max body size**. 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.

10.2.9.2. 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 v.3.1.2 and later, 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 later, 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

This tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**.

On Cribl-managed Cloud Workers (with certain plan tiers), 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 only when the Destination is in an error state.
Backpressure	After the Destination has been in a backpressure state for a specified amount of time, Cribl Stream/Edge queues and stores data to a disk until the backpressure event resolves.
Always on	Cribl Stream/Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

Max file size: 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`.

Max queue size: 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 **Max PQ size per Worker Process** on the Group Settings/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.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream/Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, switch this to `No`. When `No` is enabled, this option appears:

- **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 **Group Settings/Fleet Settings** > 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 **Output ID** has been defined.)

To learn more about persistent queue behavior and how to use it with your system, see:

- Persistent Queues
- Planning for Persistent Queues
- Configuring Persistent Queues

Processing Settings

Post-Processing

Pipeline: Pipeline 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
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).

Advanced Settings

Validate server certs: Toggle to `Yes` 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 to `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Edge to cycle through them in the order returned.

Compress: Compresses the payload body before sending. Defaults to `Yes` (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`.

Max body size (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 **Max body size** 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 **Max Body Size** 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 can cause the payload size to be smaller than the configured **Max body size**. 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, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

10.2.10. Prometheus

10.2.10.1. 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

This tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**.

On Cribl-managed Cloud Workers (with certain plan tiers), 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 only when the Destination is in an error state.
Backpressure	After the Destination has been in a backpressure state for a specified amount of time, Cribl Stream/Edge queues and stores data to a disk until the backpressure event resolves.
Always on	Cribl Stream/Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

Max file size: 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`.

Max queue size: 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 **Max PQ size per Worker Process** on the Group Settings/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.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream/Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, switch this to `No`. When `No` is enabled, this option appears:

- **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 **Group Settings/Fleet Settings** > 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 **Output ID** has been defined.)

To learn more about persistent queue behavior and how to use it with your system, see:

- Persistent Queues
- Planning for Persistent Queues

- Configuring Persistent Queues

Processing Settings

Post-Processing

Pipeline: Pipeline 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: 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 that are **not** authorized by a trusted CA (for example, the system's CA). Defaults to `Yes`.

Round-robin DNS: Toggle to `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Edge to cycle through them in the order returned.

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`.

Max body size (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 **Max body size** 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. 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. The default `Yes` value 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 assumes that payloads are snappy-compressed by default.
- 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.

10.2.10.2. 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

This tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**.

On Cribl-managed Cloud Workers (with certain plan tiers), 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 only when the Destination is in an error state.
Backpressure	After the Destination has been in a backpressure state for a specified amount of time, Cribl Stream/Edge queues and stores data to a disk until the backpressure event resolves.
Always on	Cribl Stream/Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

Max file size: 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`.

Max queue size: 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 **Max PQ size per Worker Process** on the Group Settings/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.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream/Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, switch this to `No`. When `No` is enabled, this option appears:

- **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 **Group Settings/Fleet Settings** > 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 **Output ID** has been defined.)

To learn more about persistent queue behavior and how to use it with your system, see:

- Persistent Queues
- Planning for Persistent Queues
- Configuring Persistent Queues

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 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: 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 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 `Yes`.

Round-robin DNS: Toggle to `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Edge to cycle through them in the order returned.

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`.

Max body size (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 **Max body size** 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.

Loki and Prometheus might complain about entries being delivered out of order when **Request concurrency** is set > `1` and any of **Flush period (sec)**, **Max body size (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 `Yes` 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 `__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.

10.2.10.3. 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

This tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**.

> On Cribl-managed Cloud Workers (with certain plan tiers), 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 only when the Destination is in an error state.
Backpressure	After the Destination has been in a backpressure state for a specified amount of time, Cribl Stream/Edge queues and stores data to a disk until the backpressure event resolves.
Always on	Cribl Stream/Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

Max file size: 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`.

Max queue size: 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 **Max PQ size per Worker Process** on the Group Settings/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.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream/Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, switch this to `No`. When `No` is enabled, this option appears:

- **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 **Group Settings/Fleet Settings** > 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 **Output ID** has been defined.)

To learn more about persistent queue behavior and how to use it with your system, see:

- Persistent Queues
- Planning for Persistent Queues
- Configuring Persistent Queues

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 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: 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

Round-robin DNS: Toggle to `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Edge to cycle through them in the order returned.

Validate server certs: Reject certificates that are not authorized by a trusted CA (for example, the system's CA). Defaults to `Yes`.

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`.

Max body size (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 **Max body size** 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 **Max Body Size** 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 `15`.

Extra HTTP headers: Name-value pairs to pass as additional HTTP headers. Values will be sent encrypted.

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 `Yes` 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.

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.

10.2.11. Splunk

10.2.11.1. Splunk HEC Destination

The **Splunk HEC** Destination can stream data to a Splunk HEC (HTTP Event Collector) receiver through the event endpoint. 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**: When enabled (default), lets you specify multiple Splunk HEC endpoints and load weights. With the default `No` setting, if 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.

This Destination does not support Mutual TLS (mTLS).

4. Next, you can configure the following **Optional Settings**:
   - **Exclude current host IPs**: This toggle appears when **Load balancing** is set to `Yes`. It 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.
   - **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.

If you're sending data to Splunk Cloud, Cribl recommends a maximum value of `1 MB` for **Max body size (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 to `Yes`.

To specify more receivers on new rows, click **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.
- **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 set **Enable Indexer Acknowledgement** on the Splunk token. With this setting enabled, 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.s
response: {"text":"Data channel is missing","code":10}
time: 2023-02-14T15:26:03.413Z
```

Persistent Queue Settings

This tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**.

On Cribl-managed Cloud Workers (with certain plan tiers), 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 only when the Destination is in an error state.
Backpressure	After the Destination has been in a backpressure state for a specified amount of time,

Mode	Description
	Cribl Stream/Edge queues and stores data to a disk until the backpressure event resolves.
Always on	Cribl Stream/Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

Max file size: 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`.

Max queue size: 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 **Max PQ size per Worker Process** on the Group Settings/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.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream/Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, switch this to `No`. When `No` is enabled, this option appears:

- **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 **Group Settings/Fleet Settings** > 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 **Output ID** has been defined.)

To learn more about persistent queue behavior and how to use it with your system, see:

- Persistent Queues
- Planning for Persistent Queues
- Configuring Persistent Queues

Processing Settings

Post-Processing

Pipeline: Pipeline 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
Greater than or equal to `400` and less than or equal to `500`.	Drop the request.

Status Code	Action
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

Output multi–metrics: Toggle to `Yes` 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 `Yes`.

Round-robin DNS: Toggle to `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Edge to cycle through them in the order returned. (This setting is available only when **General Settings** > **Load balancing** is set to `No`.)

Compress: Compresses the payload body before sending. Defaults to `Yes` (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`. Each request can potentially hit a different HEC receiver.

Max body size (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 **Max body size (KB)**. This upper limit is defined by the maximum content length for the default HTTP Event Collector (HEC) provided by Splunk Cloud.

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 can cause the payload size to be smaller than the configured **Max body size**. 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.

The next two fields appear only when the **General Settings** > **Load balancing** option is set to `Yes`.

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.)

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.

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.

10.2.11.2. 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

This tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**.

> On Cribl-managed Cloud Workers (with certain plan tiers), 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 only when the Destination is in an error state.
Backpressure	After the Destination has been in a backpressure state for a specified amount of time, Cribl Stream/Edge queues and stores data to a disk until the backpressure event resolves.
Always on	Cribl Stream/Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

Max file size: 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`.

Max queue size: 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 **Max PQ size per Worker Process** on the Group Settings/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.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream/Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, switch this to `No`. When `No` is enabled, this option appears:

- **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 **Group Settings/Fleet Settings** > 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 **Output ID** has been defined.)

To learn more about persistent queue behavior and how to use it with your system, see:

- Persistent Queues
- Planning for Persistent Queues
- Configuring Persistent Queues

TLS Settings (Client Side)

Enabled defaults to `No`. When toggled to `Yes`:

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 `Yes`. **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 `Yes`.

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. 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 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 to `Yes` 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 set to `Yes` (the 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 to `No`, exposes the following alternative option:

Max failed health checks: 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 **Max connections**, the throttling rate is applied per connection. For instance, if **Max connections** 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`.

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.

Log failed requests to disk: Toggling to `Yes` makes the payloads of any (future) failed requests available for inspection. See Inspecting Payloads 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.

Inspecting Payloads to Troubleshoot Closed Connections

When a downstream receiver closes connections from this Destination (or just stops responding), inspecting the payloads of the resulting failed requests 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 payloads of failed requests. 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 payloads from failed requests. At the command line, navigate to that directory and inspect any payloads that you think might be relevant.

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.

10.2.11.3. 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**: By 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). This behavior is controlled by a toggle set to `Yes` by default. Cribl strongly recommends keeping this setting enabled 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 set to its `No` 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]{#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

This tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**.

On Cribl-managed Cloud Workers (with certain plan tiers), 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 only when the Destination is in an error state.
Backpressure	After the Destination has been in a backpressure state for a specified amount of time, Cribl Stream/Edge queues and stores data to a disk until the backpressure event resolves.
Always on	Cribl Stream/Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

Max file size: 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`.

Max queue size: 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 **Max PQ size per Worker Process** on the Group Settings/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.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream/Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, switch this to `No`. When `No` is enabled, this option appears:

- **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 **Group Settings/Fleet Settings** > 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 **Output ID** has been defined.)

To learn more about persistent queue behavior and how to use it with your system, see:

- Persistent Queues
- Planning for Persistent Queues
- Configuring Persistent Queues

TLS Settings (Client Side)

Enabled defaults to `No`. When toggled to `Yes`:

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 `Yes`.

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. 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 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 to `Yes` 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 set to `Yes` (the 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 to `No`, exposes the following alternative option:

Max failed health checks: 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).

Max connections: 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.

Log failed requests to disk: Toggling to `Yes` makes the payloads of any (future) failed requests available for inspection. See Inspecting Payloads 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 **Max connections** setting is also applied per Worker Process. To continue the example, if you set **Max connections** 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.

Inspecting Payloads to Troubleshoot Closed Connections

When a downstream receiver closes connections from this Destination (or just stops responding), inspecting the payloads of the resulting failed requests 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 payloads of failed requests. 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 payloads from failed requests. At the command line, navigate to that directory and inspect any payloads that you think might be relevant.

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
```

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-splunl
```

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>_serve
```

If you are using RSA keys, instead use:

```
openssl rsa -in 100_<your-splunk-cloud>_splunkcloud/default/<your-splunk-cloud>_serv
```

Step 4. Extract the public key for the Server Certificate:

```
openssl x509 -in 100_<your-splunk-cloud>_splunkcloud/default/<your-splunk-cloud>_ser
```

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 **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.

10.2.12. 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

This tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**.

On Cribl-managed Cloud Workers (with certain plan tiers), 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 only when the Destination is in an error state.
Backpressure	After the Destination has been in a backpressure state for a specified amount of time, Cribl Stream/Edge queues and stores data to a disk until the backpressure event resolves.
Always on	Cribl Stream/Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

Max file size: 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`.

Max queue size: 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 **Max PQ size per Worker Process** on the Group Settings/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.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream/Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, switch this to `No`. When `No` is enabled, this option appears:

- **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 **Group Settings/Fleet Settings** > 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 **Output ID** has been defined.)

To learn more about persistent queue behavior and how to use it with your system, see:

- Persistent Queues
- Planning for Persistent Queues
- Configuring Persistent Queues

Processing Settings

Post-Processing

Pipeline: Pipeline 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
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: Defaults to `Yes` 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 `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Edge to cycle through them in the order returned.

Compress: Defaults to `Yes` 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.

Max body size (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.

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 can cause the payload size to be smaller than the configured **Max body size**. 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.

10.2.13. 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** to `Yes`. See the TLS Settings section for details. Defaults to `No`.
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 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.

Persistent Queue Settings

This tab is displayed when the **Backpressure behavior** is set to **Persistent Queue**.

On Cribl-managed Cloud Workers (with certain plan/license tiers), this tab exposes only the destructive **Clear Persistent Queue** button (described below in 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 Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Mode: Choose the condition for engaging persistent queues. Options include `Smart` and `Always On`.

- **Smart**: Engages persistent queues only when the Destination detects backpressure. This mode helps optimize performance by using disk storage only when necessary.
- **Always On**: Always write events into the persistent queue before forwarding them. This mode ensures maximum data durability but may require more disk space.

Max file size: 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, queueing is stopped and data blocking is applied. Enter a numeral with units of KB, MB, and so on.

Queue file path: The location for the persistent queue files. This will be of the form: `your/path/here/<worker-id>/<output-id>`. Defaults to: `$CRIBL_HOME/state/queues`.

Compression: Codec to use to compress the persisted data, once a file is closed. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue is exerting backpressure (because disk is low or at full capacity). **Block** is the same behavior as non-PQ blocking, corresponding to the **Block** option on the **Backpressure behavior** drop-down. **Drop new data** throws away incoming data, while leaving the contents of the PQ unchanged.

Clear Persistent Queue: Select this "panic" button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear - because this will free up disk space by permanently deleting the queued data, without delivering it to downstream receivers. (Appears only after **Output ID** has been defined.)

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, toggle this off. Doing so will expose this additional control:

- **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's drain rate can boost the throughput of new/active connections, by reserving more resources for them. You can further optimize Workers' startup connections and CPU load at **Fleet Settings** > Worker Processes.

Processing Settings

Post-Processing

Pipeline: Pipeline 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: 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 to `Yes` 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**, nor by another trusted CA (for example, the system's CA). Defaults to `Yes`.

Round-robin DNS: Toggle to `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Edge to cycle through them in the order returned.

Compress: Compresses the payload body before sending. Defaults to `Yes` (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`.

Max body size (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 **Max body size** 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 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.

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.

10.2.14. 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

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 how you want events formatted, either `JSON` or `Raw`.
        - `JSON` automatically structures the data in a standard JSON format.
        - `Raw` (the default) sends only the event's `_raw` value.
3. Under **Authentication** select an **Authentication method** from the drop-down:
    - **Manual**: Displays a **Next-Gen SIEM auth token** field for you to enter your authentication token.
    - **Secret**: Displays a **Next-Gen SIEM auth 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. Optionally, configure any Persistent Queue, Processing, Retries, and Advanced settings outlined in the below sections.
6. Click **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

This tab is displayed when the **Backpressure behavior** is set to **Persistent Queue**.

Max file size: 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`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Destination will stop queueing data and apply the **Queue-full** behavior. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Fleet Settings.

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.

Queue-full behavior: Whether to block or drop events when the queue is exerting backpressure (because disk is low or at full capacity). **Block** is the same behavior as non-PQ blocking, corresponding to the **Block** option on the **Backpressure behavior** drop-down. **Drop new data** throws away incoming data, while leaving the contents of the PQ unchanged.

Clear Persistent Queue: Click this "panic" button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear - because this will free up disk space by permanently deleting the queued data, without delivering it to downstream receivers. (Appears only after **Output ID** has been defined.)

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, toggle this off. Doing so will expose this additional control:

- **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's drain rate can boost the throughput of new/active connections, by reserving more resources for them. You can further optimize Workers' startup connections and CPU load at **Fleet Settings** > Worker Processes.

Processing Settings

Post-Processing

Pipeline: Pipeline 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
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).

Advanced Settings

Validate server certs: Defaults to `Yes` 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 `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Edge to cycle through them in the order returned.

Compress: Defaults to `Yes` 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.

Max body size (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.

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 can cause the payload size to be smaller than the configured **Max body size**. 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.

10.2.15. Datadog Destination

Cribl Edge can send log and metric events to Datadog. Datadog supports metrics of type `gauge`, `counter`, `rate`, and `distribution` via its REST API.

Cribl Edge sends data via two primary Datadog endpoints, which must be accessible from your Cribl environment:

- Logs: Data is routed through `http-intake.logs.{domain}`
- Metrics: Data is sent via `api.{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 (for example, `env:prod`, `env_staging:east`).
   - **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. Defaults to `info`; the drop-down offers many other severity options.

Datadog uses the above five fields (`source`, `host`, `__service`, `tags`, and `__severity`) to enhance searches.

   - **Allow API key from events**: If toggled to `Yes`, 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

This tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**.

> On Cribl-managed Cloud Workers (with certain plan tiers), 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 only when the Destination is in an error state.
Backpressure	After the Destination has been in a backpressure state for a specified amount of time, Cribl Stream/Edge queues and stores data to a disk until the backpressure event resolves.
Always on	Cribl Stream/Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

Max file size: 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`.

Max queue size: 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 **Max PQ size per Worker Process** on the Group Settings/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.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream/Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, switch this to `No`. When `No` is enabled, this option appears:

- **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 **Group Settings/Fleet Settings** > 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 **Output ID** has been defined.)

To learn more about persistent queue behavior and how to use it with your system, see:

- Persistent Queues
- Planning for Persistent Queues
- Configuring Persistent Queues

Processing Settings

Post-Processing

Pipeline: Pipeline 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
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: Toggle to `Yes` 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 to `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Edge to cycle through them in the order returned.

Compress: Compresses the payload body before sending. Defaults to `Yes` (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`.

Max body size (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 **Max body size** 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 **Max Body Size** settings.

Max events per request: 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. Otherwise, Cribl Edge will send it as a log event.

You can use these fields to override outbound event values for log events:

- `__service`
- `__severity`

No internal fields are supported for metric events.

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.

10.2.16. 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**:
    - **Dynatrace processing type**: Choose the processing type for your data in Dynatrace. This setting determines how Dynatrace will handle and analyze the incoming data.
        - `Grail with OpenPipeline`: Logs are processed using Dynatrace's Grail with OpenPipeline. This allows for advanced log processing and analysis capabilities.

        - `Grail without OpenPipeline`: Logs are stored and analyzed in Grail without the additional processing provided by OpenPipeline.
        - `Log Classic`: Uses Dynatrace's classic log monitoring capabilities without the advanced features of Grail.
        - `Other / Unknown`: This option provides flexibility for custom or non-standard processing setups in Dynatrace.
    - **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

This tab is displayed when the **Backpressure behavior** is set to **Persistent Queue**.

On Cribl-managed Cloud Workers (with certain plan/license tiers), this tab exposes only the destructive **Clear Persistent Queue** button (described below in 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 Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Max file size: 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`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Destination will stop queueing data and apply the **Queue-full** behavior. Required, and defaults to `5 GB`. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Fleet Settings.

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.

Queue-full behavior: Whether to block or drop events when the queue is exerting backpressure (because disk is low or at full capacity). **Block** is the same behavior as non-PQ blocking, corresponding to the **Block** option on the **Backpressure behavior** drop-down. **Drop new data** throws away incoming data, while leaving the contents of the PQ unchanged.

Clear Persistent Queue: Select this "panic" button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear - because this will free up disk space by permanently deleting the queued data, without delivering it to downstream receivers. (Appears only after **Output ID** has been defined.)

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, toggle this off. Doing so will expose this additional control:

- **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's drain rate can boost the throughput of new/active connections, by reserving more resources for them. You can further optimize Workers' startup connections and CPU load at **Fleet Settings** > Worker Processes.

Processing Settings

Post-Processing

Pipeline: Pipeline 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
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 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 `Yes`.

Round-robin DNS: Toggle to `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Edge to cycle through them in the order returned.

Compress: Compresses the payload body before sending. Defaults to `Yes` (recommended).

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`.

Max body size (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 **Max body size** 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 **Max Body Size** 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: 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.

10.2.17. 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

This tab is displayed when the **Backpressure behavior** is set to **Persistent Queue**.

On Cribl-managed Cloud Workers (with certain plan/license tiers), this tab exposes only the destructive **Clear Persistent Queue** button (described below in 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 Edge will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Max file size: 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, queueing is stopped and data blocking is applied. Enter a numeral with units of KB, MB, and so on.

Queue file path: The location for the persistent queue files. This will be of the form: `your/path/here/<worker-id>/<output-id>`. Defaults to: `$CRIBL_HOME/state/queues`.

Compression: Codec to use to compress the persisted data, once a file is closed. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue is exerting backpressure (because disk is low or at full capacity). **Block** is the same behavior as non-PQ blocking, corresponding to the **Block** option on the **Backpressure behavior** drop-down. **Drop new data** throws away incoming data, while leaving the contents of the PQ unchanged.

Clear Persistent Queue: Select this "panic" button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear - because this will free up disk space by permanently deleting the queued data, without delivering it to downstream receivers. (Appears only after **Output ID** has been defined.)

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, toggle this off. Doing so will expose this additional control:

- **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's drain rate can boost the throughput of new/active connections, by reserving more resources for them. You can further optimize Workers' startup connections and CPU load at **Fleet Settings** > Worker Processes.

Processing Settings

Post-Processing

Pipeline: Pipeline 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: 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 to `Yes` 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: Toggle to `Yes` 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 to `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Edge to cycle through them in the order returned.

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.

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`.

Max body size (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 **Max body size** 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. 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.

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 ServiceNow. 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.

10.2.18. 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 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`.

Max file open time (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).

Max file idle time (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`.

Max open files: 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 dirs: When toggled on (the default), 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 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).

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. The default setting (`Yes`) 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 `Yes`.

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.

10.2.19. 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**.

- **Compress**: 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 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 Working with Parquet 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 to `Yes` 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

Max file size (MB): Maximum uncompressed output file size. Files of this size will be closed and moved to final output location. Defaults to `32`.

Max file open time (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`.

Max file idle time (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`.

Max open files: 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 `Max file size (MB)` or the `Max file open time (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 set to `Yes` (the 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 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 staging dirs** is enabled. Defaults to `300` seconds (every 5 minutes). Minimum configurable interval is `10` seconds; maximum is `86400` seconds (every 24 hours).

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`.

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.

10.2.20. 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

This tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**.

> On Cribl-managed Cloud Workers (with certain plan tiers), 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 only when the Destination is in an error state.
Backpressure	After the Destination has been in a backpressure state for a specified amount of time, Cribl Stream/Edge queues and stores data to a disk until the backpressure event resolves.
Always on	Cribl Stream/Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

Max file size: 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`.

Max queue size: 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 **Max PQ size per Worker Process** on the Group Settings/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.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream/Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, switch this to `No`. When `No` is enabled, this option appears:

- **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 **Group Settings/Fleet Settings** > 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 **Output ID** has been defined.)

To learn more about persistent queue behavior and how to use it with your system, see:

- Persistent Queues
- Planning for Persistent Queues
- Configuring Persistent Queues

Processing Settings

Post-Processing

Pipeline: Pipeline 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
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: Toggle to `Yes` 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 to `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Edge to cycle through them in the order returned.

Compress: Compresses the payload body before sending. Defaults to `Yes` (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`.

Max body size (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 **Max body size** 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. 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.

10.2.21. 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. (For example, `http://localhost:8086/write`.)
    - **Use v2 API**: You can enable the InfluxDB v2 API with InfluxDB version 1.8 or later. This toggle defaults to `No` – which falls back to the v1 API, and displays a **Database** field.

    If you toggle **Use v2 API** to `Yes`, 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. (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 enabled, 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). Defaults to `Yes`.

- **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

This tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**.

On Cribl-managed Cloud Workers (with certain plan tiers), 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 only when the Destination is in an error state.
Backpressure	After the Destination has been in a backpressure state for a specified amount of time, Cribl Stream/Edge queues and stores data to a disk until the backpressure event resolves.
Always on	Cribl Stream/Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

Max file size: 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`.

Max queue size: 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 **Max PQ size per Worker Process** on the Group Settings/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.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream/Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, switch this to `No`. When `No` is enabled, this option appears:

- **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 **Group Settings/Fleet Settings** > 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 **Output ID** has been defined.)

To learn more about persistent queue behavior and how to use it with your system, see:

- Persistent Queues
- Planning for Persistent Queues
- Configuring Persistent Queues

Processing Settings

Post-Processing

Pipeline: Pipeline 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
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 that are **not** authorized by a trusted CA (for example, the system's CA). Defaults to `Yes`.

Round-robin DNS: Toggle to `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Edge to cycle through them in the order returned.

Compress: Compresses the payload body before sending. Defaults to `Yes` (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 before blocking. This is set per Worker Process. Defaults to `5`.

Max body size (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 **Max body size** 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. 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, helping you identify delivery issues. Analyze the graphs showing events and bytes in/out over time.

10.2.22. 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).

   - **Compress**: 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. **IAM Roles for Amazon EC2**: Loaded from AWS Identity and Access Management (IAM) roles attached to an EC2 instance.
2. **Shared Credentials File**: Loaded from the shared credentials file (`~/.aws/credentials`).
3. **Environment Variables**: Loaded from environment variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`.
4. **JSON File on Disk**: Loaded from a JSON file on disk.
5. **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 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 Working with Parquet 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 to `Yes` 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

Max file size (MB): Maximum uncompressed output file size. Files of this size will be closed and moved to final output location. Defaults to `32`.

Max file open time (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`.

Max file idle time (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`.

Max open files: 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 `Max file size (MB)` or the `Max file open time (sec)` conditions is met.

Max concurrent file parts: 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 set to `Yes` (the 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 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 staging dirs** is enabled. Defaults to `300` seconds (every 5 minutes). Minimum configurable interval is `10` seconds; maximum is `86400` seconds (every 24 hours).

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`.

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. The default setting (`Yes`) 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 `Yes`.

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
```

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.

10.2.23. NetFlow Destination

Cribl Edge supports passing through NetFlow v5 and v9 UDP traffic to NetFlow Collectors.

Type: Non-Streaming | TLS Support: No | PQ Support: Yes

Configurie Cribl Edge to Output to NetFlow

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 NetFlow definition. If you clone this Destination, Cribl Edge will add `-CLONE` to the original **Output ID**.
    - **Description**: Optionally, enter a description.
    - **NetFlow Destinations**: Add the downstream NetFlow Collectors to which Cribl Edge should send data.
    - **Address:** Hostname or IP address of the Collector.
    - **Port:** Port number to connect to on the Collector. Defaults to `2055`, which is the standard port for NetFlow traffic.
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 to process data before sending the data out using this output.

Advanced Settings

DNS resolution period (sec): Re-resolve any hostnames after each interval of this many seconds, and pick up destinations from records. Defaults to `0` seconds. A value of `0` means every datagram sent will incur a DNS lookup. A non-zero value improves performance but can reduce the overall reliability if the DNS records for the downstream Collectors change frequently.

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:

- `__netflowRaw` : Directly routes the raw NetFlow packet data to a downstream NetFlow Collector.

Considerations for Working with NetFlow Data FlowSets

For both NetFlow v5 and v9, Cribl Edge:

- Can forwards NetFlow packets to other NetFlow Collectors. However, it cannot modify the the contents of the incoming packet. In other words, Cribl Edge forwards the packets verbatim as they came in.
- Only routes NetFlow packets from upstream Exporters and cannot generate its own NetFlow packets.
- Cannot send non-NetFlow input data to NetFlow Collectors.

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.

10.2.24. 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

In Cribl Edge 4.1 and later, 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 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. 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 Defaults to `No`. When toggled to `Yes`:

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 `Yes`.

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.

Persistent Queue Settings

This tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**.

On Cribl-managed Cloud Workers (with certain plan tiers), 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 only when the Destination is in an error state.
Backpressure	After the Destination has been in a backpressure state for a specified amount of time, Cribl Stream/Edge queues and stores data to a disk until the backpressure event resolves.
Always on	Cribl Stream/Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

Max file size: 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`.

Max queue size: 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 **Max PQ size per Worker Process** on the Group Settings/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.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream/Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, switch this to `No`. When `No` is enabled, this option appears:

- **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 **Group Settings/Fleet Settings** > 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 **Output ID** has been defined.)

To learn more about persistent queue behavior and how to use it with your system, see:

- Persistent Queues
- Planning for Persistent Queues
- Configuring Persistent Queues

Processing Settings

Post-Processing

Pipeline: Pipeline 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: 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

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`.

Max body size (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 **Max body size** 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 to `Yes` 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 to `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Edge to cycle through them in the order returned.

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.

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.

10.2.25. SentinelOne DataSet Destination

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

This tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**.

> On Cribl-managed Cloud Workers (with certain plan tiers), 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 only when the Destination is in an error state.
Backpressure	After the Destination has been in a backpressure state for a specified amount of time, Cribl Stream/Edge queues and stores data to a disk until the backpressure event resolves.
Always on	Cribl Stream/Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

Max file size: 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`.

Max queue size: 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 **Max PQ size per Worker Process** on the Group Settings/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.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream/Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, switch this to `No`. When `No` is enabled, this option appears:

- **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 **Group Settings/Fleet Settings** > 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 **Output ID** has been defined.)

To learn more about persistent queue behavior and how to use it with your system, see:

- Persistent Queues
- Planning for Persistent Queues
- Configuring Persistent Queues

Processing Settings

Post-Processing

Pipeline: Pipeline 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
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, `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 `Yes` 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 to `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Edge to cycle through them in the order returned.

Compress: Compresses the payload body before sending. Defaults to `Yes` (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 before blocking. This is set per Worker Process. Defaults to `5`.

Max body size (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 **Max body size** 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 **Max Body Size** 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**.

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.

10.2.26. 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 Defaults to `No`. When toggled to `Yes`:

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 `Yes`.

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.

Persistent Queue Settings

This tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**.

On Cribl-managed Cloud Workers (with certain plan tiers), 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 only when the Destination is in an error state.
Backpressure	After the Destination has been in a backpressure state for a specified amount of time, Cribl Stream/Edge queues and stores data to a disk until the backpressure event resolves.
Always on	Cribl Stream/Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

Max file size: 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`.

Max queue size: 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 **Max PQ size per Worker Process** on the Group Settings/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.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream/Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, switch this to `No`. When `No` is enabled, this option appears:

- **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 **Group Settings/Fleet Settings** > 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 **Output ID** has been defined.)

To learn more about persistent queue behavior and how to use it with your system, see:

- Persistent Queues
- Planning for Persistent Queues
- Configuring Persistent Queues

Processing Settings

Post-Processing

Pipeline: Pipeline 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: 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 to `Yes` 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

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`.

Max body size (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 **Max body size** 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 to `Yes` 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 to `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Edge to cycle through them in the order returned.

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.

10.2.27. 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

This tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**.

> On Cribl-managed Cloud Workers (with certain plan tiers), 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 only when the Destination is in an error state.
Backpressure	After the Destination has been in a backpressure state for a specified amount of time, Cribl Stream/Edge queues and stores data to a disk until the backpressure event resolves.
Always on	Cribl Stream/Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

Max file size: 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`.

Max queue size: 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 **Max PQ size per Worker Process** on the Group Settings/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.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream/Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, switch this to `No`. When `No` is enabled, this option appears:

- **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 **Group Settings/Fleet Settings** > 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 **Output ID** has been defined.)

To learn more about persistent queue behavior and how to use it with your system, see:

- Persistent Queues
- Planning for Persistent Queues
- Configuring Persistent Queues

Processing Settings

Post-Processing

Pipeline: Pipeline 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
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: Toggle to `Yes` 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 to `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Edge to cycle through them in the order returned.

Compress: Compresses the payload body before sending. Defaults to `Yes` (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`.

Max body size (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 **Max body size** 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 can cause the payload size to be smaller than the configured **Max body size**. 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.

10.2.28. 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 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.

10.2.29. 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

This tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**.

On Cribl-managed Cloud Workers (with certain plan tiers), 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 only when the Destination is in an error state.
Backpressure	After the Destination has been in a backpressure state for a specified amount of time, Cribl Stream/Edge queues and stores data to a disk until the backpressure event resolves.
Always on	Cribl Stream/Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

Max file size: 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`.

Max queue size: 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 **Max PQ size per Worker Process** on the Group Settings/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.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream/Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, switch this to `No`. When `No` is enabled, this option appears:

- **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 **Group Settings/Fleet Settings** > 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 **Output ID** has been defined.)

To learn more about persistent queue behavior and how to use it with your system, see:

- Persistent Queues
- Planning for Persistent Queues
- Configuring Persistent Queues

Processing Settings

Post-Processing

Pipeline: Pipeline 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
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: Toggle to `Yes` 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 to `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Edge to cycle through them in the order returned.

Compress: Compresses the payload body before sending. Defaults to `Yes` (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`.

Max body size (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 **Max body size** 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 **Max Body Size** 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: 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**.

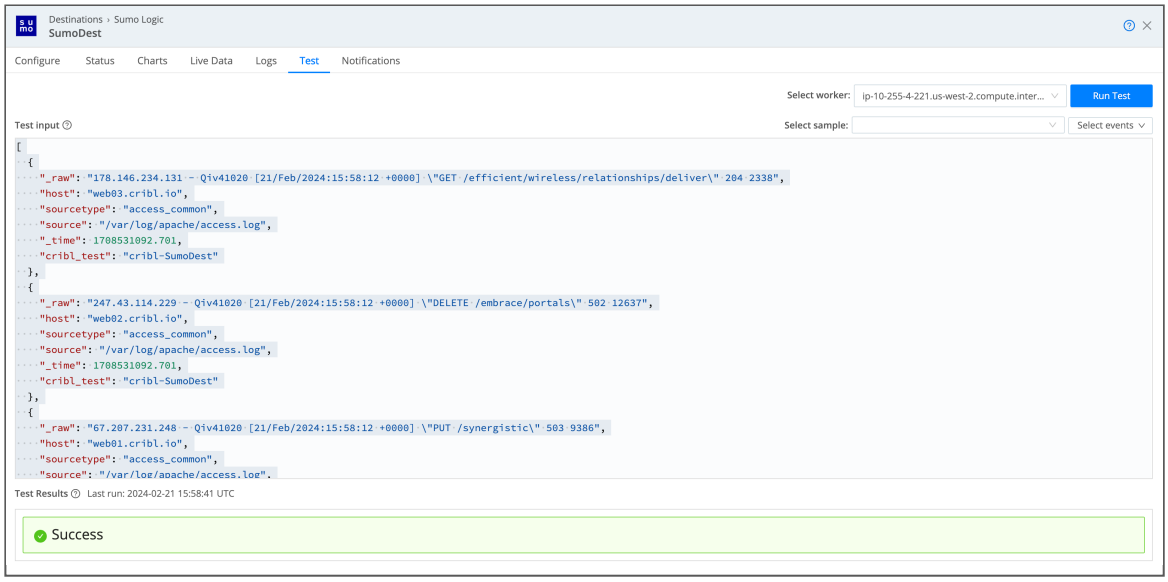
Safe headers: Add headers to declare them as safe to log in plaintext. (Sensitive headers such as `authorization` will always be redacted, even if listed here.) Use a tab or hard return to separate header names.

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

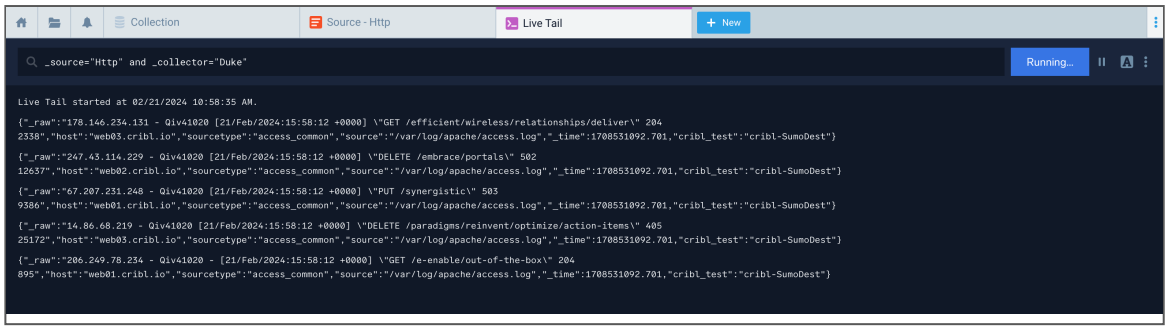
To verify data flow, we'll use the Destination's **Test** feature while running a **Live Tail** session in Sumo Logic.

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 nodes.

10.2.30. 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.

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`. When enabled (default), lets you 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`.

Enabling the **Load balancing** toggle replaces the static **General Settings** > **Address** and **Port** fields with the following controls:

Exclude current host IPs: This toggle appears when **Load balancing** is set to `Yes`. It 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.

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 **Max connections**, the throttling rate is applied per connection. For instance, if **Max connections** 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

This tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**.

On Cribl-managed Cloud Workers (with certain plan tiers), 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 only when the Destination is in an error state.
Backpressure	After the Destination has been in a backpressure state for a specified amount of time, Cribl Stream/Edge queues and stores data to a disk until the backpressure event resolves.
Always on	Cribl Stream/Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

Max file size: 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`.

Max queue size: 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 **Max PQ size per Worker Process** on the Group Settings/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.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream/Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, switch this to `No`. When `No` is enabled, this option appears:

- **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 **Group Settings/Fleet Settings** > 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 **Output ID** has been defined.)

To learn more about persistent queue behavior and how to use it with your system, see:

- Persistent Queues
- Planning for Persistent Queues
- Configuring Persistent Queues

TLS Settings (Client Side)

Enabled defaults to `No`. When toggled to `Yes`:

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 `Yes`.

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

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 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.

Log failed requests to disk: Toggling to `Yes` makes the payloads of any (future) failed requests available for inspection. See Inspect Payloads 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.

Max connections: 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.

Inspect Payloads to Troubleshoot Closed Connections

When a downstream receiver closes connections from this Destination (or just stops responding), inspecting the payloads of the resulting failed requests 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 payloads of failed requests. 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 payloads from failed requests. At the command line, navigate to that directory and inspect any payloads that you think might be relevant.

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:

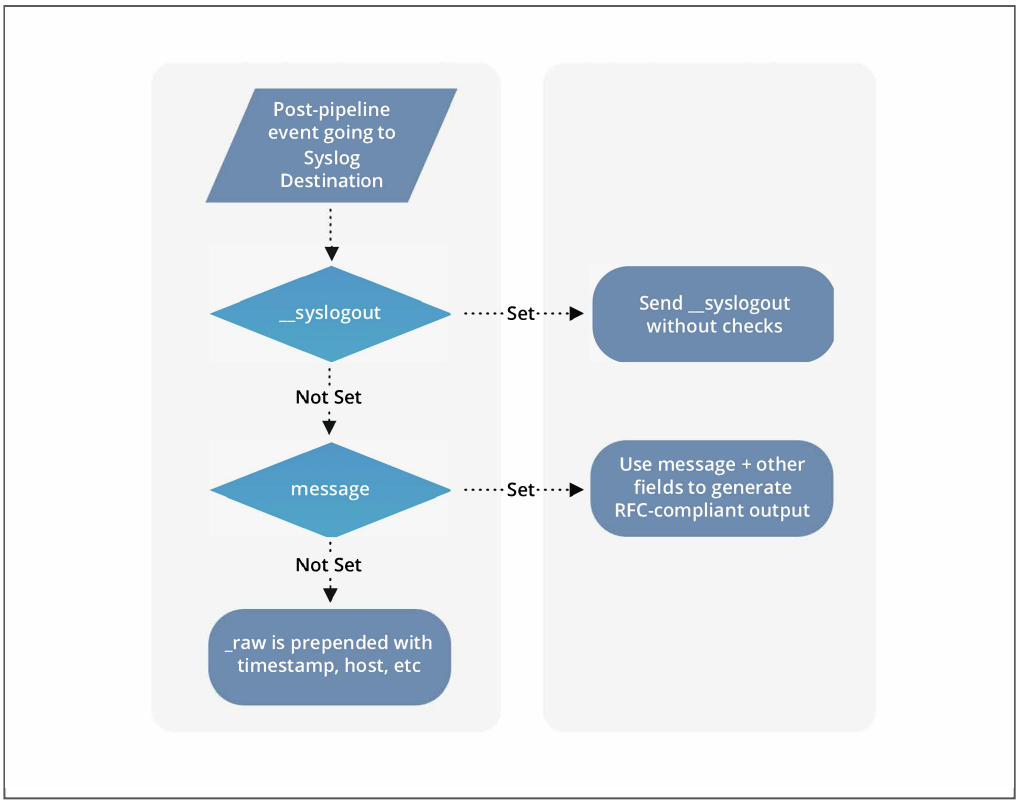
- `<214>1 2022-07-11T18:58:45.000Z testbox testapp 42 - [exampleSDID@32473 iut="3" foo="bar" this="that"] Here is a message`

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:

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.

10.2.31. 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**: When enabled (default), lets you specify multiple destinations. For details, see Load Balancing Settings below.
      The following two fields appear **only** with the `No` setting.
    - **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**: This toggle appears when **Load balancing** is set to `Yes`. It 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.

- **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 **Max connections**, the throttling rate is applied per connection. For instance, if **Max connections** 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

This tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**.

> On Cribl-managed Cloud Workers (with certain plan tiers), 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 only when the Destination is in an error state.
Backpressure	After the Destination has been in a backpressure state for a specified amount of time, Cribl Stream/Edge queues and stores data to a disk until the backpressure event resolves.
Always on	Cribl Stream/Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

Max file size: 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`.

Max queue size: 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 **Max PQ size per Worker Process** on the Group Settings/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.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream/Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, switch this to `No`. When `No` is enabled, this option appears:

- **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 **Group Settings/Fleet Settings** > 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 **Output ID** has been defined.)

To learn more about persistent queue behavior and how to use it with your system, see:

- Persistent Queues
- Planning for Persistent Queues
- Configuring Persistent Queues

TLS Settings (Client Side)

Use TLS defaults to `No`. When toggled to `Yes`:

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 `Yes`.

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 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: Toggling to `Yes` makes the payloads of any (future) failed requests available for inspection. See Inspecting Payloads 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.

Setting General Settings > **Load balancing** to `Yes` adds 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.)

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.

Inspecting Payloads to Troubleshoot Closed Connections

When a downstream receiver closes connections from this Destination (or just stops responding), inspecting the payloads of the resulting failed requests 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 payloads of failed requests. 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 payloads from failed requests. At the command line, navigate to that directory and inspect any payloads that you think might be relevant.

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.

10.2.32. 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

This tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**.

> On Cribl-managed Cloud Workers (with certain plan tiers), 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 only when the Destination is in an error state.
Backpressure	After the Destination has been in a backpressure state for a specified amount of time, Cribl Stream/Edge queues and stores data to a disk until the backpressure event resolves.
Always on	Cribl Stream/Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

Max file size: 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`.

Max queue size: 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 **Max PQ size per Worker Process** on the Group Settings/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.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream/Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, switch this to `No`. When `No` is enabled, this option appears:

- **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 **Group Settings/Fleet Settings** > 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 **Output ID** has been defined.)

To learn more about persistent queue behavior and how to use it with your system, see:

- Persistent Queues
- Planning for Persistent Queues
- Configuring Persistent Queues

Processing Settings

Post-Processing

Pipeline: Pipeline 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
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: Toggle to `Yes` 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 to `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Edge to cycle through them in the order returned.

Compress: Compresses the payload body before sending. Defaults to `Yes` (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`.

Max body size (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 **Max body size** 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 **Max body size**. 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.

10.2.33. 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**: When enabled (default), lets you specify multiple Webhook URLs and load weights. With the default `No` setting, if 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**: This toggle appears when **Load balancing** is set to `Yes`. It 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.
    - **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 Defaults to `No`. When toggled to `Yes`:

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.

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

This tab displays when the **Backpressure behavior** option in **General settings** is set to **Persistent Queue**.

> On Cribl-managed Cloud Workers (with certain plan tiers), 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 only when the Destination is in an error state.
Backpressure	After the Destination has been in a backpressure state for a specified amount of time, Cribl Stream/Edge queues and stores data to a disk until the backpressure event resolves.
Always on	Cribl Stream/Edge immediately queues and stores all data on a disk for all events, even when there is no backpressure.

Max file size: 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`.

Max queue size: 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 **Max PQ size per Worker Process** on the Group Settings/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.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream/Edge will send earlier queued events before forwarding newly arrived events. To instead prioritize new events before draining the queue, switch this to `No`. When `No` is enabled, this option appears:

- **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 **Group Settings/Fleet Settings** > 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 **Output ID** has been defined.)

To learn more about persistent queue behavior and how to use it with your system, see:

- Persistent Queues
- Planning for Persistent Queues
- Configuring Persistent Queues

Processing Settings

Post-Processing

Pipeline: Pipeline 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
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 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 `Yes`.

Round-robin DNS: Toggle to `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Edge to cycle through them in the order returned.

Compress: Compresses the payload body before sending. Defaults to `Yes` (recommended).

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`.

Max body size (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 **Max body size** 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 **Max Body Size** 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: 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}, ${fieldB}` `` (with literal backticks). Defaults to `__httpOut` – that is, the value of the `__httpOut` field. Use the button at right to open a validation modal.
- **Drop when null**: If toggled to `Yes`, Cribl Edge will drop events when the **Source expression** evaluates to `null`.
- **Event delimiter**: Delimiter string to insert between events. Defaults to the newline character (`\n`). Cannot be a space (this will be converted to `\n`).
- **Content type**: Content type to use for requests. Defaults to `application/x-ndjson`. Any content types set in **Advanced Settings > Extra HTTP headers** will override this entry.
- **Batch expression**: Expression specifying how to format the payload for each batch. Defines a wrapper object in which to include the formatted events, such as a JSON document. This enables requests to APIs that require such objects. To reference the events to send, use the `${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"
      }
   ]
}
```

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 }
```

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 Examples

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 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;
}
```

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** > **Max events per request**.

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
}
```

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
```

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)}]`
```

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> }
```

Here's an example of an event formatted for Bulk API:

```
{ "index" : { "_index" : "test", "_id" : "1" } }
{ "field1" : "value1", "field2" "value2", "field3" : "value3" }
```

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":"
```

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`;
```

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

- 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.

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.

10.3. Destination Backpressure Triggers

This topic 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 does so to notify the Sources generating data that its buffers are full and that the incoming data flow should be paused 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 – i.e., 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**.

HTTP-based Destinations include:

- Splunk HEC
- Elasticsearch
- Wavefront
- SignalFx
- Sumo Logic
- Honeycomb
- Datadog
- NewRelic Ingest Logs & Metrics
- NewRelic Ingest Events
- Microsoft Sentinel
- CloudWatch Logs
- Webhook
- Grafana Cloud
- Prometheus
- Loki
- Cribl HTTP
- InfluxDB
- Google Cloud Chronicle
- Google Cloud Logging

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 – i.e., 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 – i.e., 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.

Filesystem-based Destinations include:

- Filesystem
- Azure Blob Storage
- Google Cloud Storage
- Amazon S3
- MinIO
- CrowdStrike Falcon LogScale
- Amazon Security Lake
- Amazon Data Lakes S3

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.

The following Destinations are TCP-based and adhere to the rules described above. 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:

- Splunk Single Instance
- Splunk Load-Balanced
- TCP JSON
- Cribl TCP
- Syslog (TCP or UDP)
- StatsD (TCP or UDP)
- StatsD Extended (TCP or UCP)
- Graphite (TCP or UCP)

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 – i.e., 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.

Cribl Edge supports the following Kafka-based Destinations:

- Kafka
- Azure Event Hubs
- Confluent Cloud
- MSK

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`.
- **Max retries**: 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 **Max retries** 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.

The following Destinations support UDP. Except for SNMP Trap, these Destinations also support TCP, but the UDP versions do not trigger backpressure.

- SNMP Trap
- Syslog
- StatsD
- StatsD Extended

- Graphite

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.

HTTP Destinations that support load balancing include:

- Splunk HEC
- Elasticsearch
- Cribl HTTP

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.

TCP load-balanced Destinations include:

- Splunk Load-Balanced
- TCP JSON
- Cribl TCP
- Syslog (TCP version only)

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.

10.4. Persistent Queues

Cribl Edge's persistent queues (PQ) supplement in-memory queues to prevent data loss during a backpressure event. When a backpressure event occurs, Cribl Edge can engage a backpressure queue and write data to disk for the duration of the outage. When the outage resolves, Cribl Edge then forwards data from the queue.

Cribl Edge's persistent queue (PQ) feature helps minimize data loss if:

- A downstream Destination receiver is unreachable.
- A downstream Destination receiver is slow to respond.
- During backpressure from Cribl Edge internal processing on configured Sources.

Persistent queues are best in environments that need more durability and reliability, such as systems where:

- Outages last longer than memory queues can sustain.
- Upstream senders do not support backpressure, such as ephemeral or network senders.

Persistent queues can be enabled:

- On Push Sources.
- On Streaming Destinations.

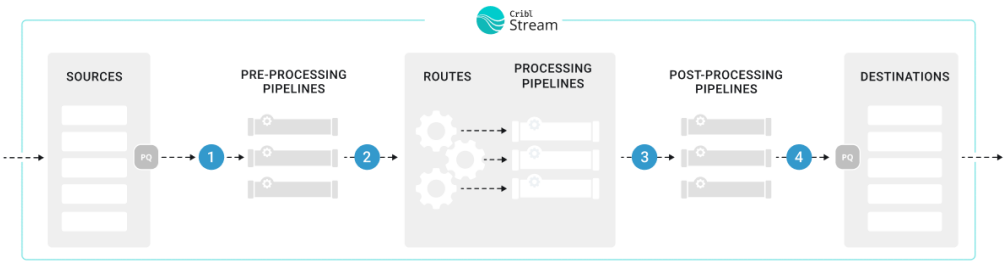
Sources can also take advantage of a Destination's queue to keep data flowing. See Supported Sources and Supported Destinations for more information.

The Persistent queue feature is only available for certain Cribl licenses. See the Pricing page for details.

How Persistent Queues Work

Backpressure events occur 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 and Destinations.

When persistent queue engages, it writes and stores data on a local file system in your environment until the backpressure event resolves.



You can enable persistent queues at both ends of the data flow

How Destinations Handle Backpressure

Destinations handle data rates using the following order of operations:

1. Under normal conditions, Destinations store data on 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 there is an inbound burst of data, a Worker Process stores events in Destination buffers. It 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. If the Destination buffer becomes too full, the Destination triggers a backpressure event.
4. During a backpressure event, Cribl Edge will respond in one of the following ways, based on the Destination **Backpressure behavior** settings:

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 events, 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 the persistent queue behavior engages, Cribl Edge will respond in one of the following ways, based on its configured backpressure mode:

Mode	Description
Error	Queues and stores data on a disk only when the Destination is in an error state.
Backpressure	After the Destination has been in a backpressure state for a specified amount of time, Cribl Edge queues and stores data to a disk 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 the amount of data stored in memory and what happens to data when the queue reaches its capacity limit.

6. By default, after data flow is re-established, a Destination will forward events in first in, first out (FIFO) order. It will send out earlier queued events before newly arriving events. However, in Cribl Edge 4.1 and later, you can instead prioritize new events by disabling Strict ordering on a Destination.

How Sources Handle Backpressure

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 two 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.

Once enabled on a specific Source, in `Smart` mode, PQ will engage when any of the following occur:

- A connected Destination is fully down, and Destination-side PQ is not enabled.
- A connected Destination is slow, or blocking.
- A downstream Cribl Pipeline/Route is slow.

These triggers are not relevant if you've enabled Source PQ in `Always On` mode. In that case, it's always on.

When PQ engages on a Source:

- Events queue *before* any pre-processing Pipeline takes effect.
- Metrics (data in) will show `0` bytes/events while events are buffered to the disk queue.

- Cribl Edge Pipelines/Routes will process the data with a delay, when PQ later drains.

Because PQ always carries some latency penalty, it is redundant to enable PQ on a Source whose upstream sender is configured to safeguard events in its own disk buffer.

If you turn off Source PQ (or the whole Source) before its queues drain, any events still queued on disk will be orphaned.

Persistent Queue Details and Constraints

Persistent queues:

- Are implemented at the Worker Process level, with independent sizing configuration and dynamic engagement per Worker Process.
- Are not infinite in size. If data cannot be delivered out, your system might run out of disk space.
- Cannot fully protect data in cases of application failure. For example, if a crash occurs, in-memory data might get lost.
- Cannot fully protect data in cases of hardware failure, such as disk failure, corruption, or machine/host loss.
- TLS-encrypted only for data in flight, and only on Destinations where TLS is supported and enabled. In an on-prem deployment, to encrypt data at rest, including disk writes/reads, you must configure encryption on the underlying storage volume(s). In Cribl.Cloud, persistent queue storage keeps the data encrypted on disk.

Cribl Edge begins to drain persistent queues when at least one receiver can accept data.

Beyond these failure scenarios, if you turn off PQ on a Source or Destination (or shut down the whole Source/Destination) before the disk queue has drained, the queued events will be orphaned 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 blocking data flow. A single live receiver will prevent persistent queue from engaging on the corresponding Destination. PQ 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), PQs engage when the brokers can't be reached.
- On HTTP/S-based Destinations, controlled to prevent retries of failed events when the buffer drains. (Some events in transit to the Destination might still emerge as duplicates.)

Supported Sources

Cribl Edge supports persistent queue for Push Sources. These sources ingest streaming data, which makes it possible to enable persistent queue.

Persistent queue 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:

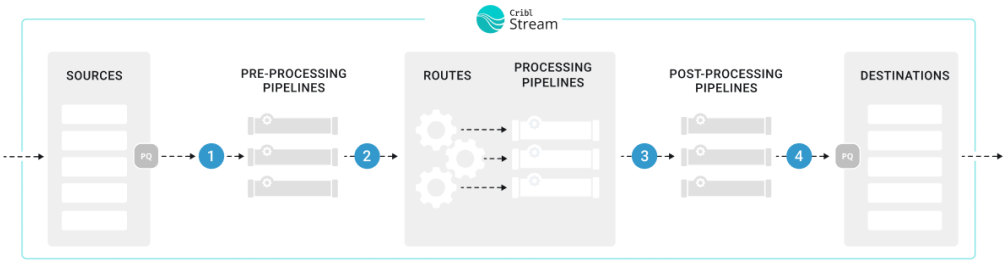
- HTTP-based Destinations
- TCP Load-Balanced Destinations
- Destinations that write to a single receiver (without load balancing enabled)

To determine whether Cribl Edge supports persistent queue for a specific Destination, consult the documentation for that Destination.

10.4.1. Planning For Persistent Queues

Use this page to guide your decision making as you decide how to handle data backpressure in your system and whether implementing persistent queue is the right strategy. This page explains:

- Use cases for implementing persistent queues (PQ).
- Details about persistent queue behavior on Sources and Destinations, with various configuration and tuning options.
- Choices you should make before enabling persistent queue.
- The implications of those choices for your system's requirements and performance.



You can enable persistent queues at both ends of the data flow

Better Practices When Implementing Persistent Queues

In general, these factors should guide your decisions about persistent queues in on-prem environments:

- The likelihood of backpressure events that could result in data loss.
- Your business requirements and your organization's level of tolerance for data loss.
- 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 and devote Worker processes to the persistent queue. Depending on your settings, persistent queues can be resource intensive.

When planning and tuning the performance of your persistent queues setting, start by monitoring your system performance and throughput and Worker processes capacity. Pay attention to metrics such CPU utilization. If needed, you can teleport into a single Worker and view all the processes and their individual CPU utilization.

After you implement a change in your system, continue to monitor the impacts of that change. Tune it as needed using the considerations discussed in the rest of this guide.

Source-Side Persistent Queue

Source-Side persistent queue can serve as a stopgap for upstream senders that don't provide their own buffering, or whose small buffers cannot queue for very long. This applies to several syslog senders and HTTP-based Sources.

Persistent queue prevents data loss when the Source is unable to handle backpressure for an extended period. Backpressure might be triggered by error conditions, or by degraded performance (slowdowns or short backups), in downstream Cribl resources or outside receivers.

Always On Versus Smart Mode

You should consider enabling the persistent queue for Push Sources in situations where:

- Senders do not provide their own buffering.
- Destination-Side PQ is not enabled.

In this situation use either `Smart` or `Always On` mode:

- `Always On` mode is a good match for highly valuable data (such as security data), and for data ingested via UDP. (Relevant UDP Sources are Raw UDP, SNMP Trap, Metrics, and Syslog with UDP enabled.)
- `Smart` mode might be sufficient for lower-grade data, such as events that are overwhelmingly `debug`-level.

Always On Requirements

`Always On` mode provides the most reliable data delivery, but you should plan for its impacts on throughput, memory, and hosts' disk resources.

Throughput

Each Source configured in `Always On` mode imposes a significant performance penalty on Workers' data ingestion rate, as compared to Sources without PQ enabled. This primarily involves the overhead to write to the filesystem. As a starting point, assume that each event will incur approximately 1 second of added latency.

Memory

`Always On` mode can also increase memory demands, as compared to Sources without PQ enabled. If memory is a constraint, you can experiment with tuning the **Max buffer size** in each Source's UI, or the

`maxBufferSize` configuration key. However, smaller buffers will reduce the efficiency of disk writes, adding more latency.

Disk

`Always On` also imposes more 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 Worker Processes (evenly weighted), 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.

Compression

Enabling the PQ compression option conserves disk space, with a compression ratio of about 8:1. But you get these savings at the cost of worsening the performance impact to throughput. Navigate this trade-off based on which factor is most valuable in your deployment.

Smart Mode Details

`Smart` mode engages only when a Source receives an unhealthy signal from a downstream process, such as when a Pipeline or Destination is down or blocked. When engaged, `Smart` mode's latency penalty is about the same as with `Always On` mode. But this latency is highly variable, because `Smart` mode engages intermittently. So expect less latency overall, with less predictability with some spikes and valleys.

`Smart` mode's CPU performance penalty is about twice that of `Always On`, because of the overhead to switch queueing on and off. But again, this is highly variable, because performance will degrade only when PQ engages/disengages. As with `Always On`, if memory is a constraint, you can experiment with reducing **Max buffer size** (UI) or `maxBufferSize` (config-file) values. Compression is supported in `Smart` mode, as in `Always On` modes.

Smart Mode Interaction with Multiple Destinations

If you enable `Smart` mode on a Source that's routed to multiple Destinations, you should plan around the unintended interaction that PQ can cause among these Destinations:

If at least one connected Destination sends an unhealthy signal, this Source will queue new events to disk even if some peer Destinations remain up. This will delay data delivery to the peer healthy Destinations. The same

interaction can occur between Destinations grouped in a connected Output Router.

To prevent this unintended behavior, you can either:

- Disable PQ on the Source, and instead configure PQ on individual Destinations.

Or

- Configure separate Worker Groups to connect this Source to separate Destinations.

Destination-Side Persistent Queue

Destination-side persistent queue can engage when the Destination is either slow or completely unavailable.

Consider enabling persistent queue on supported Destinations when:

- You need to safeguard against data loss during extended errors.
- You need to smooth out CPU load spikes when a Destination recovers from a transient outage.
- You send data to receivers that have intermittent outages, such as Splunk or Chronicle.

Backpressure Modes

When persistent queue is enabled, Destinations have three available modes for responding to slow or unavailable Destinations:

Mode	Description
Error	Queues and stores data on a disk only when the Destination is in an error state.
Backpressure	After the Destination has been in a backpressure state for a specified amount of time, Cribl Edge queues and stores data to a disk 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 the amount of data stored in memory and what happens to data when the queue reaches its capacity limit.

Consider using `Always On` if you need to prevent data loss and ensure that you retain all data. This mode will queue and store data for all events. However, be aware that choosing `Always On` could impact your system performance because writing and reading data to disk is resource-intensive. You might experience slower system throughput or notice that your Worker Nodes are over-utilized or regularly run at full capacity.

Always size your disk capacity according to your risk tolerance. Cribl's initial recommendation is to size for at least 1 day's worth of storage on each Worker Node.

Fallback (Queue-Full) Behavior

Unlike Source PQ, on Destinations where you've enabled PQ, you also select a fallback **Queue-full behavior**.

With the default `Block` option, if a Destination's persistent queue fills up, the Destination will block back to the corresponding Source. (Remember that a Destination can be fed by multiple Sources.) If the Source has its own PQ option enabled, it will begin queueing data. If not, the block signal will go all the way back to connected senders.

If you instead set the **Queue-full behavior** to `Drop new data`, Cribl Edge will simply discard data intended for the unavailable Destination, but will still allow the data to proceed to parallel Destinations.

Good use cases for `Block`: A simple Cribl Edge Route relaying Universal Forwarder data to a Splunk receiver; or, any Destination where you absolutely **must not** lose data.

Destination PQ Better Practices

- Cribl generally recommends enabling the PQ **Compression** option on Destinations. (Disable compression if you find that it adds excessive latency.)
- When draining large volumes after recovery from an outage, you might find that PQ is crushing its Destination. In this case, disable **Strict ordering**, and tune the **Drain rate limit (EPS)** to maintain reasonable throughput for new data.

Better Practices for Source and/or Destination PQ

If your Source(s) and Destination(s) both support PQ, which side should you enable?

Enabling Source-side PQ (in `Always On` mode) is the most versatile and reliable choice. Source PQ is further upstream than Destination PQ, and can safeguard data bound for multiple Destinations.

However, you might opt for Destination-side PQ if your senders provide their own buffering, or if you don't want to add the system resources that Source Always On mode requires.

Enabling **both** Source- and Destination-side PQ on a Route is likely overkill. It can also complicate diagnosing issues.

Warning for Auto-Scaling Environments

With **persistent** storage, events on disk survive a Cribl Stream/Edge or OS restart. But you cannot count on this with ephemeral/auto-scaling environments. Stream Workers must **not** be destroyed while data has not drained from a PQ.

PQ Status Notifications and Monitoring

Depending on your Cribl license, you may be able to configure Notifications that will be sent when PQ files exceed a configurable percentage of allocated storage, or when a Destination reaches the queue-full state described above.

These notifications always appear in Cribl Edge's UI and internal logs. You can also send them to external systems. For setup details, see Destination-State Notifications.

Destination PQ Notifications

With Destination PQ enabled, you can configure Notifications around two trigger conditions. Cribl recommends that you create both:

- Destination Backpressure Activated. This will send Notifications whenever backpressure engages, causing blocked or dropped events. It will also send Notifications when a queue fills up, falling back to blocked or dropped events.
- Persistent Queue Usage (saturation). Note that you can set these Notifications for **multiple** saturation levels.

Source PQ Alerting

Currently, Cribl Edge does not support Notifications around Source PQ states. However, if you've enabled `Always On` mode, alerts that PQ has engaged would be pointless – because all of the Source's incoming data is automatically written to the disk queue.

Also, if desired: On Fleets or customer-managed (hybrid or on-prem) Worker Groups, you can enable the internal CriblLogs Source, create a Route to forward Cribl's internal logs to a downstream service of your choice, and configure alerts from there.

Monitoring PQ

Cribl Edge's native Monitoring dashboards interact with PQ in particular ways. When PQ engages, assume that your sole sources of truth about Sources' or Destinations' throughput are the dashboards at:

- **Monitoring** > **System** > **Queues (Sources),** and
- **Monitoring** > **System** > **Queues (Destinations)**.

Other Monitoring resources become unreliable during queueing. These include the top-level **Monitoring** dashboard, and:

- **Monitoring** > **Data** > **Sources,** or
- **Monitoring** > **Data** > **Destinations**.

These will not accurately represent queued data because, once PQ engages, data will be queued to disk, rather than arriving into Pipelines and downstream resources (with Source PQ) or into the affected Destinations (with Destination PQ). While the data is queueing, the two above dashboards for:

- Affected Sources will show no events per second (EPS) or bytes ingested while their data is queueing.
- Affected Destinations will show EPS and bytes sent while their data is queueing, but these metrics can be misleading. Here, they measure data going into the queue, but not proceeding to downstream receivers.

Some other Monitoring quirks to keep in mind:

- Activity in the **Monitoring** > **System** > **Queues (Sources)** and **Monitoring** > **System** > **Queues (Destinations)** dashboards sometimes shows up only when queues drain.
- Cribl Edge currently does not provide a Monitoring dashboard for disk queues' own volumes.

These gaps provide another reason to enable PQ Notifications where available.

Remember that with **or** without PQ enabled, Monitoring dashboards do not show usage of integrations' in-memory buffers.

After the unhealthy condition resolves, and disk queues start to drain, dashboards will resume accurately representing delivery rates and volumes.

Metrics and Capture Limitations

Finally, while PQ provides many benefits, also note the following limitations:

- While events go to a Source-side queue, Cribl Edge does not record metrics about that Source's inbound data (events/second or bytes).
- While events go to a Destination-side queue, metrics reflect events/second and bytes going into that Destination's queue, not proceeding beyond Cribl Edge.
- Data captures might not work while a Destination is blocked.

10.4.2. Configuring Persistent Queues

The following sections outline configuring PQ for Cribl.Cloud Workers managed by Cribl, and then for on-prem and hybrid Cribl.Cloud Workers.

Configuring PQ for Cribl-Managed Workers in Cribl.Cloud

Cribl may support persistent queuing Cribl-managed Workers in Cribl.Cloud deployments, depending on your Cribl license. Configuration is considerably simpler with Cribl-managed Workers in Cribl.Cloud than with customer-managed (hybrid or on-prem) Workers (both covered below). You configure PQ individually on each Source and Destination that supports it.

Sources

1. Select a Source to open its configuration modal.
2. Select the **Persistent Queue Settings** tab.
3. Toggle **Enable Persistent Queue** to on.

On Cribl-managed Workers in Cribl.Cloud, this tab exposes only the **Enable Persistent Queue** toggle. When enabled, PQ is automatically configured in `Always On` mode, with a maximum queue size of 1 GB of data per PQ-enabled Source, per Worker Process. The 1 GB limit is on uncompressed inbound data, and there is no compression applied to the queue.

These options are not configurable. If you want to finely configure the maximum queue size, compression, PQ mode, and other options, use a hybrid Group.

Destinations

1. Select a Destination to open its configuration modal.
2. Set **Backpressure behavior** to `Persistent Queueing`.

On Cribl-managed Workers in Cribl.Cloud, the resulting **Persistent Queue Settings** tab exposes only a **Clear Persistent Queue** button. Once enabled, PQ is automatically configured with a maximum queue size of 1 GB of uncompressed data per PQ-enabled Destination, per Worker Process. The 1 GB limit is on uncompressed outbound data, and no compression is applied to the queue. If the queue fills up, Cribl Edge will block data flow.

These options are not configurable. If you want to finely configure the maximum queue size, compression, queue-full fallback behavior, and other options, use a hybrid Group.

If you want to disable persistent queuing:

1. Wait for the queue to fully drain. (The **Clear Persistent Queue** button is available as a destructive fallback. This will free up disk space, but will do so by deleting the queued data **without** sending it on to receivers.)
2. In the config modal's **General Settings**, select a different **Backpressure behavior**.
3. Re-save the config.

Logs

Cribl internal logs for Cribl-managed Workers in Cribl.Cloud might include entries labeled `Revived PQ Worker Process`. These indicate where a Cloud 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.

Configuring PQ for On-Prem and Hybrid Workers

Cribl may support persistent queuing may for Workers in on-prem deployments and for hybrid Cribl.Cloud Workers, depending on your Cribl license. (Cribl-managed Workers in Cribl.Cloud have a simpler configuration.) You configure persistent queueing individually for each Source and Destination that supports it.

On Sources:

1. Select a Source to open its configuration modal.
2. Select **Persistent Queue Settings**.
3. Toggle **Enable Persistent Queue** to on.

On Destinations:

1. Select a Destination to open its configuration modal.
2. Set **Backpressure behavior** to `Persistent Queueing`.

These selections expose the additional controls outlined below.

For an example of optimizing several PQ throttling options listed below, see our How Does Persistent Queuing Work Inside Cribl Stream? blog post.

Source-Side PQ Only

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 Cribl Edge's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Edge's data processing engine.

Max buffer size: 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 per connection, not just per Worker Process – and this can dramatically expand memory usage.)

Commit frequency: The number of events to send downstream before committing that Stream has read them. Defaults to `42`.

Always On versus Smart Mode

In Cribl Edge 4.1 and later, Source-side PQ defaults to `Always On` mode when you configure a new Source. `Always On` offers the highest data durability, by minimizing backpressure and potential data loss on senders. However, this mode can slightly slow down data throughput, and can require you to provision more machines and/or faster disks.

If you are upgrading from a pre-4.1 version where `Smart` mode was the default, this change does not affect your existing Sources' configurations. However:

- If you create new Stream Sources programmatically, and you want to enforce the previous `Smart` mode, you'll need to update your existing code.
- You can optimize Workers' startup connections and CPU load at **Fleet Settings** > Worker Processes.

Consider switching to `Always On` mode if you find `Smart` mode causing Source-side PQ to frequently engage, and you've configured PQ with a large **Max file size** (e.g., 1 GB). However, a second option here is to retain `Smart` mode, but reduce the **Max file size** to a value no higher than the default `1 MB`.

(Source-side PQ excessively triggers backpressure only when it reaches the last queued file **and** that file is large. Keeping this threshold low prevents that condition.)

Common PQ Settings (Source and Destination Sides)

Max file size: 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 Edge applies the default `1 MB`.

Max queue size: The maximum amount of disk space that Cribl allowa the queue to consume, on each Worker Process. Once th queue reaches this limit, this Source or Destination will stop queueing data; Sources

will block, and Destinations will apply your configured **Queue-full behavior**. Defaults to `5 GB`. Enter a numeral with units of KB, MB, GB, or TB. For details, see Optimizing Max Queue Size.

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 closes. Defaults to `None`; `Gzip` is also available.

Optimizing Max Queue Size

> ⚠ When you upgrade Fleets, and customer-managed (hybrid or on-prem) Worker Groups, to Cribl Edge 4.4 and later: Any undefined PQ **Max queue size** values in pre-4.4 Source/Destination configs will be set to the new 5 GB default value. This is a guardrail to control queues' consumption of host machines' disk space.

Adjust this default to match your needs. If some of your integrations' persistent queues had an undefined **Max queue size**, queues larger than 5 GB will trigger PQ fallback behavior until you resize this limit.

Also, in Cribl Edge 4.4 and later, you can define this field's highest accepted value in the new **Group Settings** > **General Settings** > **Limits** > **Storage** > **Max PQ size per Worker Process** field. The corresponding `limits.yml` key is `maxPQSize`. Consult Cribl Support before increasing that limit's 1 TB default.

The changes here do not affect any existing config with a defined **Max queue size**, and do not affect the nonconfigurable 1 GB **Max queue size** for Cribl-managed Workers in Cribl.Cloud.

Max Queue Size with Compression

If you enable **Compression** and also enter a **Max queue size** limit, be aware of how these options interact:

- Cribl Edge currently applies the **Max queue size** limit against the **uncompressed** data volume entering the queue.
- With this combination, Cribl recommends that you set the **Max queue size** limit **higher** than the volume's total available disk space – again disregarding compression. This will maximize queue saturation and minimize data loss. (For details, see Known Issues.)

Destination-Side PQ Only

This section lists PQ configuration options on Cribl Edge Destinations.

Queue-Full Behavior

During a backpressure event, Cribl Edge will respond in one of the following ways, based on the Destination **Backpressure behavior** settings:

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 events, 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.

If the persistent queue behavior engages, Cribl Edge will respond in one of the following ways, based on its configured backpressure mode:

Mode	Description
Error	Queues and stores data on a disk only when the Destination is in an error state.
Backpressure	After the Destination has been in a backpressure state for a specified amount of time, Cribl Edge queues and stores data to a disk 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 the amount of data stored in memory and what happens to data when the queue reaches its capacity limit.

Clear Persistent Queue

Select this “panic” button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear - because this will free up disk space by permanently deleting the queued data, without delivering it to downstream receivers. (Appears only after you define the **Output ID**.)

This section's remaining controls are available in Cribl Edge 4.1 and later.

Strict Ordering

The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Edge will send earlier queued events before forwarding newly arrived events. Disabling Strict ordering prioritizes processing new events over those currently residing in the PQ. When you toggle this setting to off, it exposes the **Drain rate limit (EPS)** control.

Using Drain Limits

Disabling **Strict ordering** setting exposes the **Drain rate limit (EPS)** field. This setting allows you to throttle the rate at which events are drained from the PQ to the destination. The default `0` value disables throttling. This setting ensures that Cribl Edge can keep up with the incoming stream of live data and doesn't get bogged down processing events from the PQ. Additionally, throttling the queue's drain rate can boost the throughput of new/active connections, by reserving more resources for them.

The Drain rate limit (EPS) is set in Events Per Second (EPS). To determine an appropriate value, follow the steps below.

Find the Worker Process EPS

1. From Cribl Edge's top nav, select **Monitoring** to access multiple dashboards.
2. The initial view shows aggregate data for all Fleets and all @{nodes}. Use the drop-downs at the upper right to isolate a single Edge Node.
3. At the upper right, change the display's granularity from the default `15 min` to `1 day`.
4. Under **Events In and Out**, locate **THROUGHPUT IN (AVG)** and copy the EPS value. (Example: `31,000 EPS`)

Calculate Drain Rate Limit

1. Cribl recommends setting the **Drain Rate Limit** to 10% of the Worker Process EPS to prioritize live data.
2. Divide the Worker Process EPS by the number of Worker Processes (WPs).
3. Calculate 10% of the result from step 2. (Example: `31,000 EPS / 14 WPs = 2,200 EPS per WP; 10% of 2,200 EPS = 220 EPS`).

Enable and Set Drain Rate Limit

1. In the PQ settings, disable **Strict ordering**. This will expose the **Drain rate limit (EPS)** field.
2. Set the **Drain rate limit (EPS)** to the calculated value from step 2 (e.g., `220 EPS`).

Monitor and Adjust

Monitor the PQ drain rate. If the queue is draining too slowly, you can carefully increase the **Drain rate limit (EPS)**. However, be cautious not to set it too high, as this could impact processing live data streams.

You can further optimize Workers' startup connections and CPU load at **Fleet Settings** > Worker Processes.

Minimum Free Disk Space

For queuing to operate properly, you must provide sufficient disk space. In distributed mode, you configure the minimum disk space to support queues (and other features) on each Fleet, at **Fleet Settings** > **General Settings** > **Limits** > **Min free disk space**. If available disk space falls below this threshold, Cribl Edge will stop maintaining persistent queues, and data loss will begin. The default minimum is 5 GB.

10.5. 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.

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.

Example

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 be dispensed
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 be dispensed
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. I.e., 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 be dispensed + stats carried forward = 200 + 60 + 40 = 300`. Number of events per each destination (weighed): `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 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.

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.

10.6. Cribl Edge To Cribl Stream

Cribl Edge automatically discovers logs, metrics, application data, etc. – 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 connected to the same Leader, without incurring additional cost.

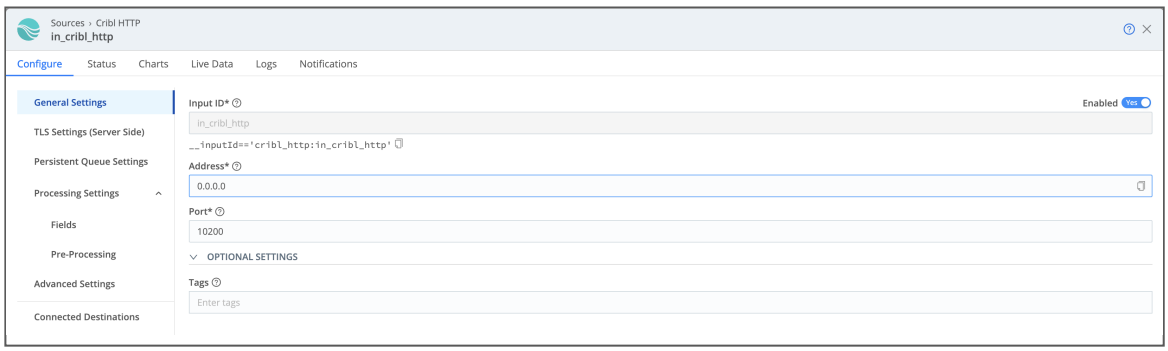
This guide outlines how to route data from an Edge Node (or an entire Fleet) to an existing Stream Worker Group for additional processing. We will walk you through the following:

- 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 Worker/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.

Configure the Exec Source on Cribl Edge

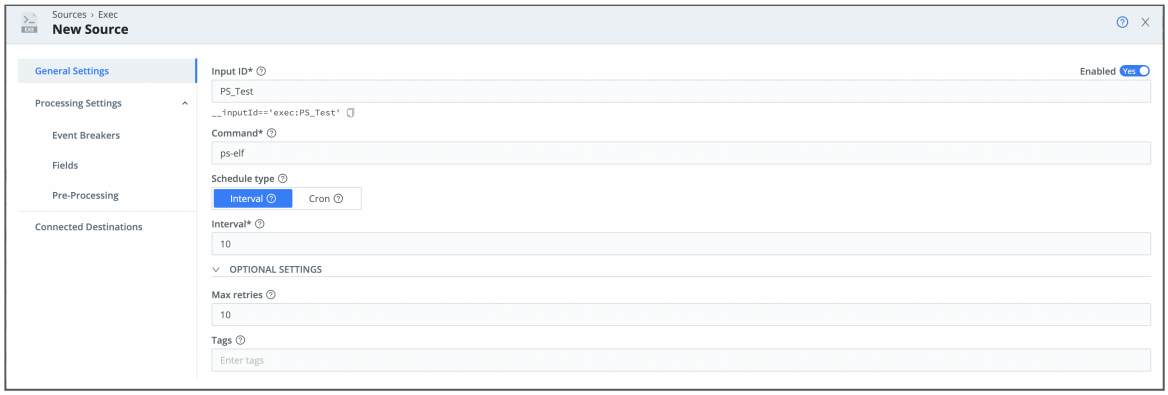
Next, we'll configure the Exec Source on your Edge Node. This Source will break the incoming streams of data into discrete events, and send them to Cribl Stream.

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.

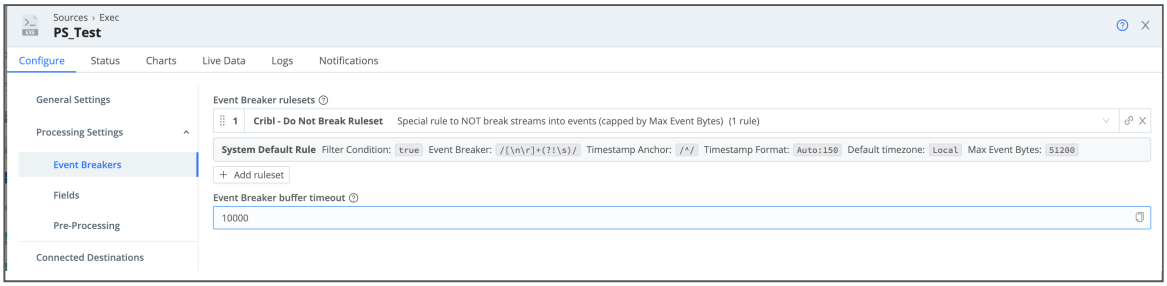
In our example, we are running the `ps` command to list and retrieve running processes every `10` seconds.



Configuring an Exec Source

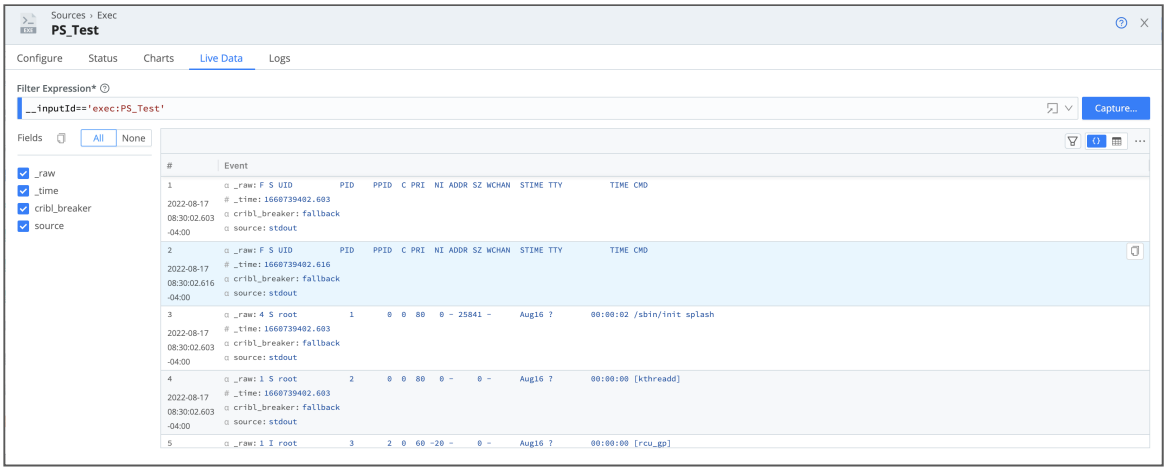
If we don't configure an Event Breaker, then with each capture we run on the dataset, each process will be ingested as its own event, without the header information. So to structure the data, we'll add an Event Breaker.

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:

- This Destination's Cribl endpoint field must point to the **Address** and **Port** you've configured on its peer Cribl HTTP Source you configured in the first step.
- The Cribl HTTP Destination must be on a Node that is connected to the same Leader as the Cribl HTTP Source(s).
- 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's **Data** submenu, to isolate the throughput on your Source.

10.7. Integrating With Other Services

10.7.1. Cribl Edge As A Sidecar Container In AWS ECS

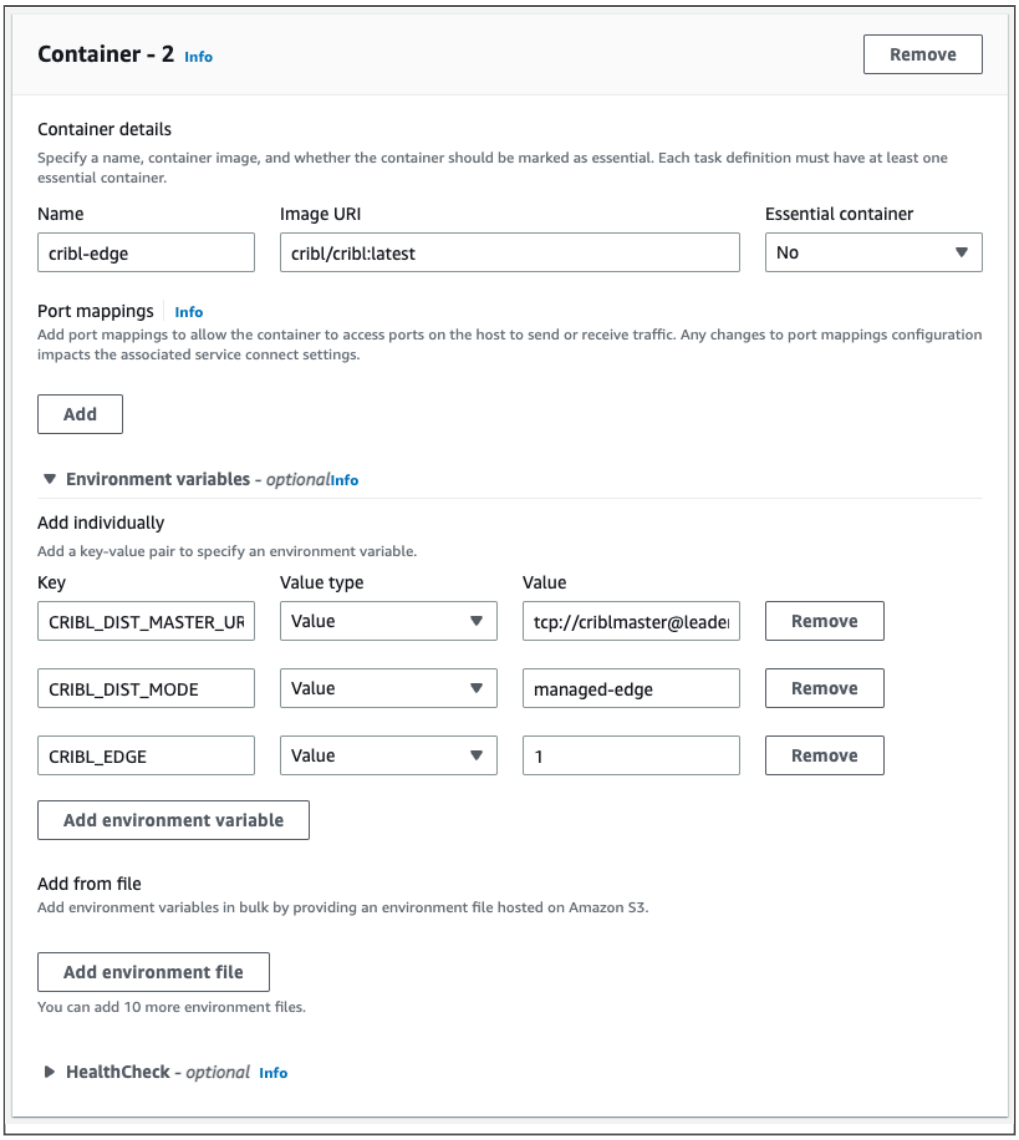
You can use Cribl Edge to collect logs from a service or task running in ECS. This is an alternative to forwarding logs to CloudWatch Logs.

Currently, you can't configure Windows 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_MASTER_URL`	`tcp://<token>@<leader>:4200`
`CRIBL_DIST_MODE`	`managed-edge`
`CRIBL_EDGE`	`1`

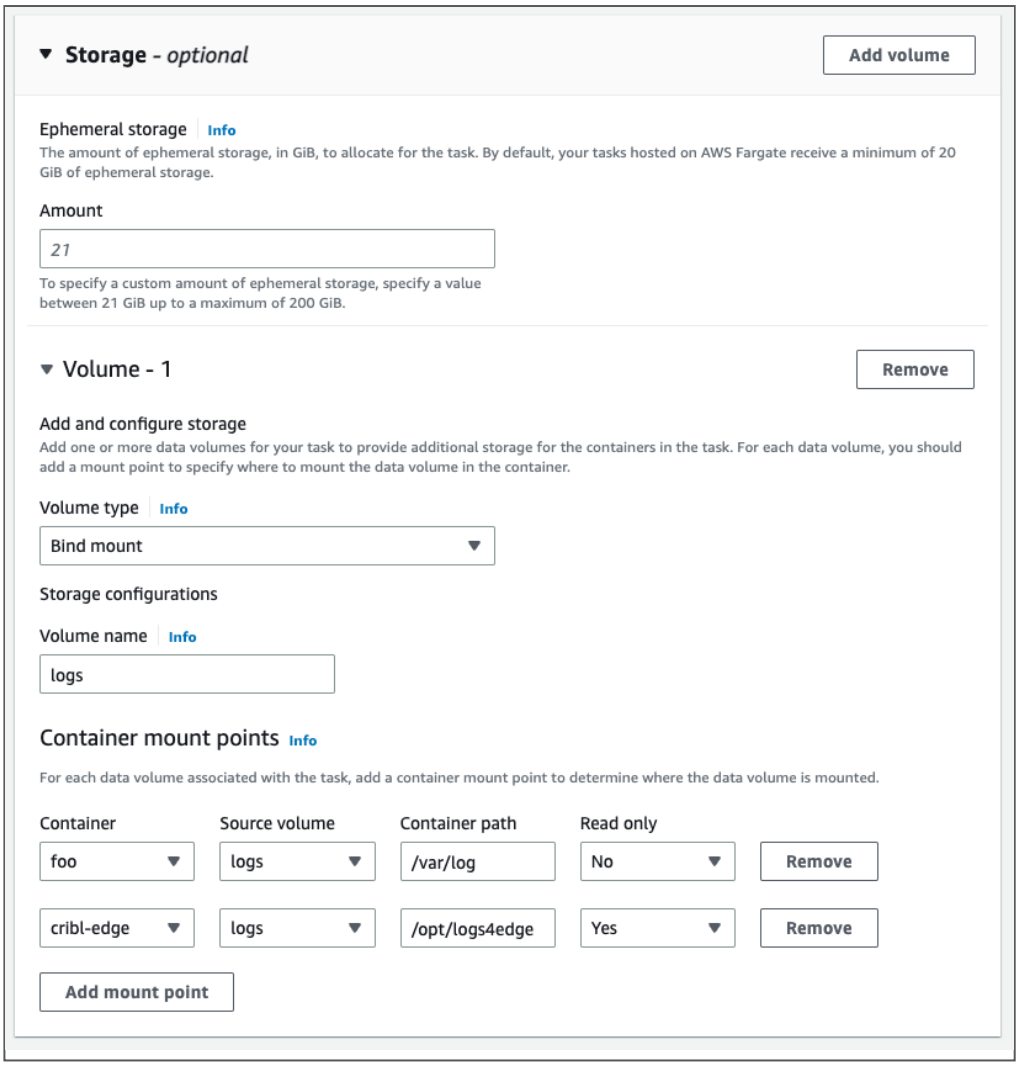


Add Environment Variables to Container Details

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.

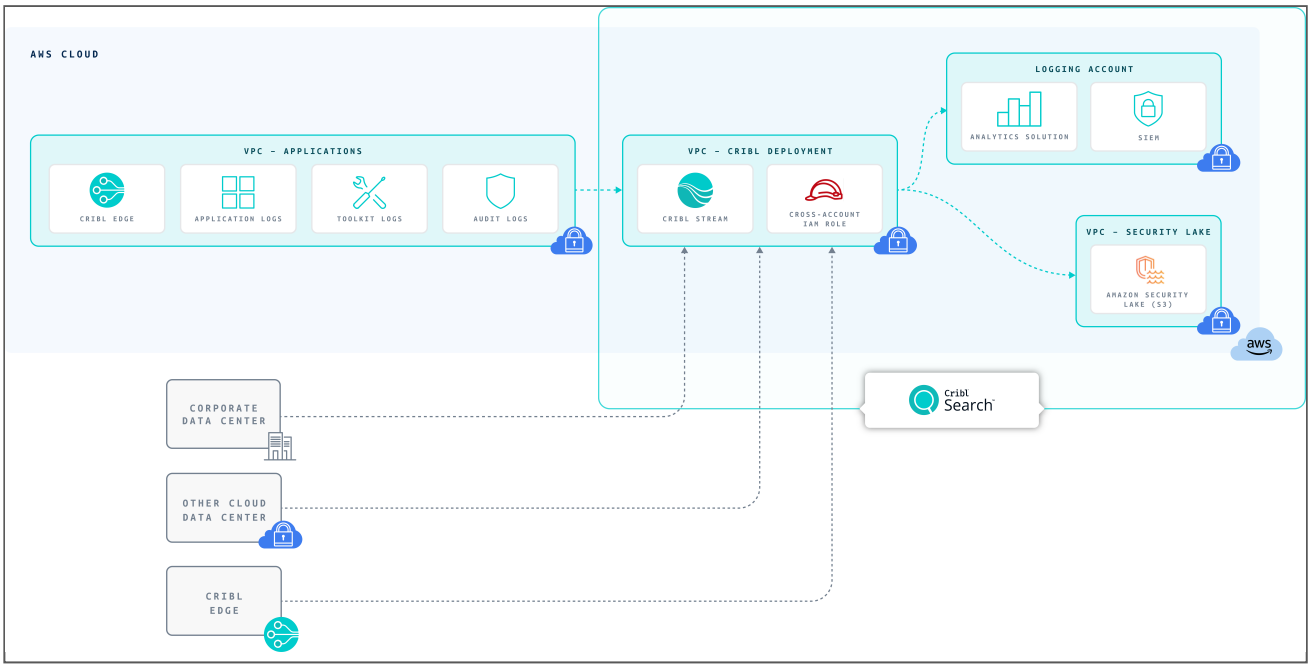
```
{
    "containerDefinitions": [
        {
            "name": "whatever",
            "image": "foo/bar:latest",
            ...
            "mountPoints": [
                {
                    "sourceVolume": "logs4edge",
                    "containerPath": "/var/log",
                    "readOnly": 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"
                },
                {
                    "name": "CRIBL_DIST_MASTER_URL",
                    "value": "tcp://cribledge@localhost:4200"
                `
```

```
                }
            ],
            "mountPoints": [
                {
                    "sourceVolume": "logs4edge",
                    "containerPath": "/opt/logs4edge",
                    "readOnly": true
                }
            ],
            "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"
                }
            }
        }
    ],
    "volumes": [
        {
            "name": "logs4edge

            ",
            "host": {}
        }
    ],
   ... truncated ...
}
```

10.7.2. 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
   ```

5. For the **New Pack ID**, enter:

   ```
   Cribl-OCSF_Post_Processor
   ```

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 here.

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.

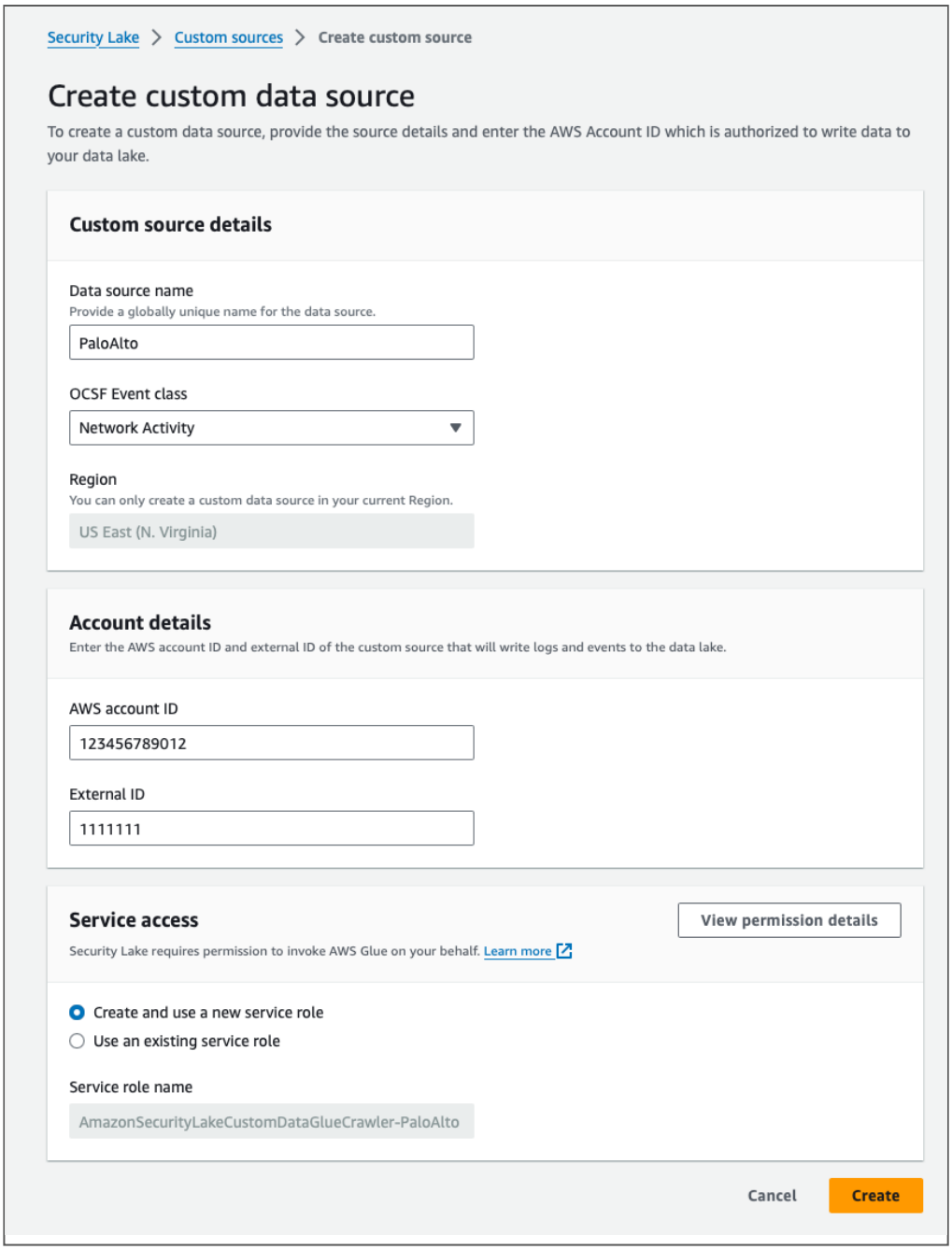
Create an Amazon Security Lake Custom Source

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 Stream, 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. Setting an invalid region 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)
```

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 pre-configured `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
```

With a Cribl.Cloud Enterprise plan, generalize the above URL's `default.main` substring to `<group-name>.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`

Data Source	OCSF Event Classes Handled	Cribl Source	Cribl Pack Display Name	Parquet Schema(s)
				`OCSF 4001 Network Activity`
Windows Logon Activity	3002	Splunk TCP	Splunk Forwarder Windows Classic Events to OCSF	`OCSF 3002 Authentication`
ZScaler Firewall and Weblogs	4001	Syslog	OCSF Post Processor Pack	`OCSF 4001 Network Activity`

10.7.3. Amazon Elastic Kubernetes Services (EKS) Add-On For Edge

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.

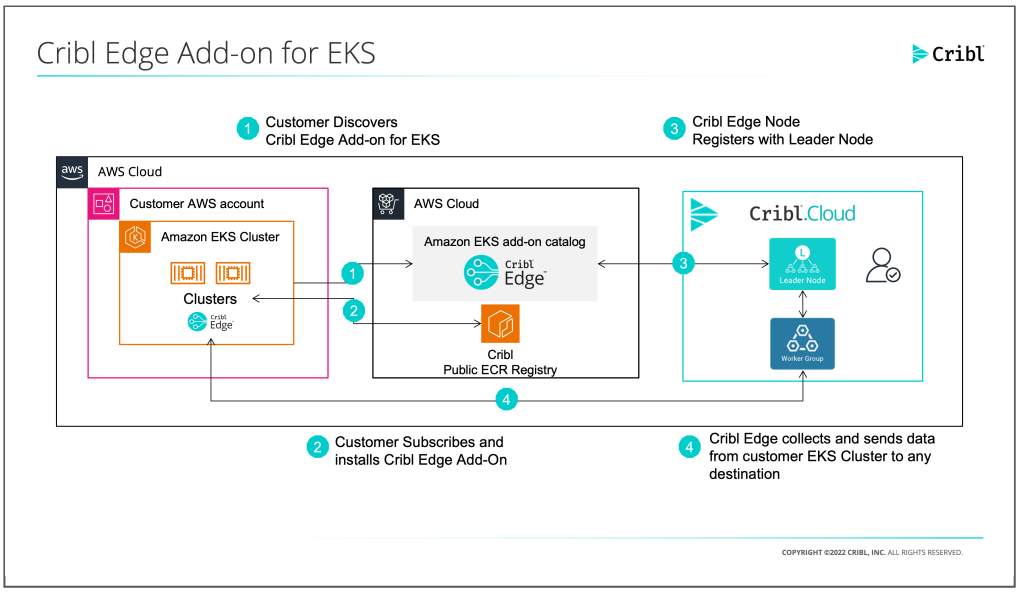
You can use your existing AWS support agreement to obtain support.

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



Amazon EKS architecture on Edge

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
 }
}
```

5. Replace the `leader` value with the URI for your Edge Leader 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/c
        "createdAt": "2022-12-01T12:18:26.497000-08:00",
        "modifiedAt": "2022-12-01T12:50:52.222000-08:00",
        "tags": {}
    }
}
```


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_cribledg
        "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://<c
    }
}
```

Disable the Cribl Edge add-on

To disable the Cribl Edge add-on, you can run the following command:

```
aws eks delete-addon --addon-name cribl_cribledge --cluster-name $YOUR_CLUSTER_NAME –region $AWS_REGION
```

More Information

If you need more help, see Amazon EKS Add-Ons documentation.

11. Monitoring Health And Metrics

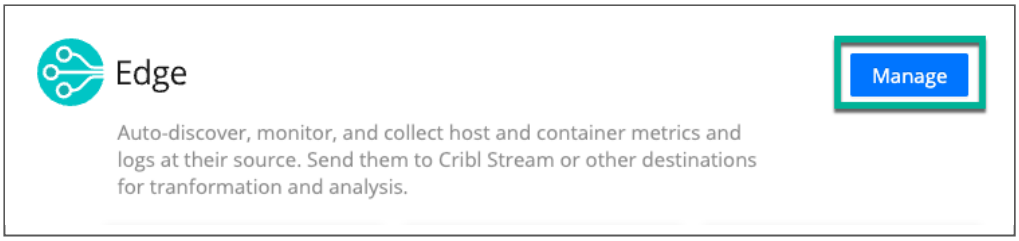
Cribl Edge offers you multiple ways to monitor your deployments in order to get an operational view of the health of your Fleets, Edge Nodes, Sources and Destinations, and more.

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 into Cribl, you'll see the Cribl Edge tile on the **Cribl Home** page. This tile displays basic Edge configuration details, including number of Fleets, Subfleets, Edge Nodes, and events and bytes over time.

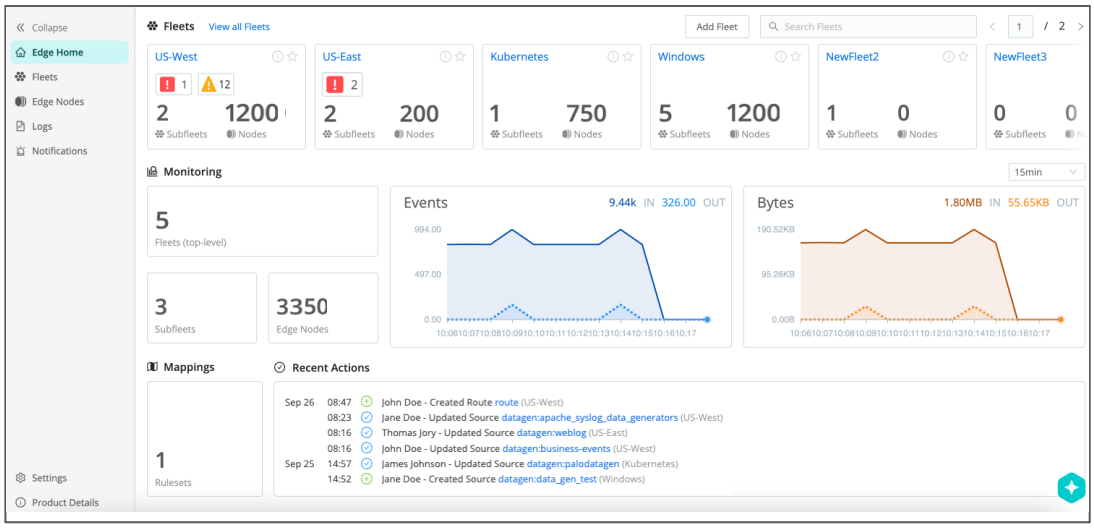


Manage Edge Tile

To monitor health and metrics of your Cribl Edge deployment, select **Manage**.

Monitor Health on the Edge Home Page

Once you select **Manage**, you'll be on the **Edge Home** page. This page shows aggregate data for all Fleets, Subfleets and Edge Nodes. The charts display information about traffic in and out of the system.



Cribl Edge Home page

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).

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.

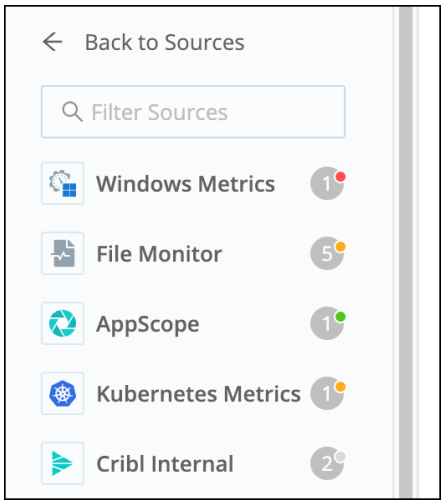
Overall Source and Destination Health

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:

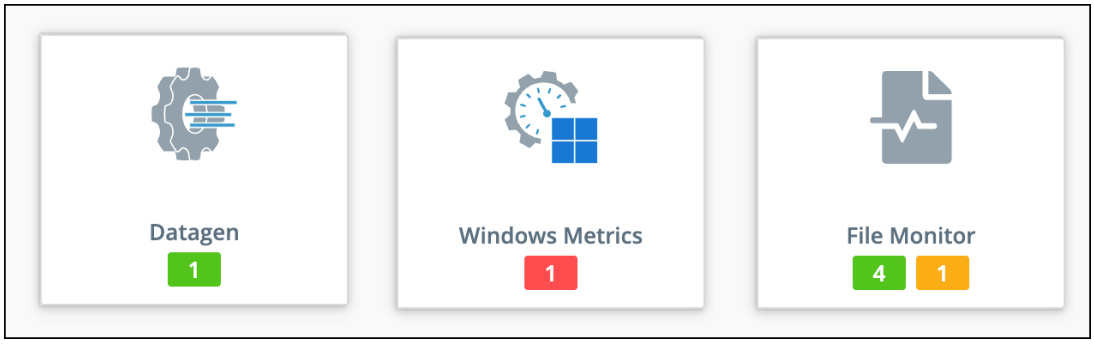
1. Select a **Fleet**.
2. Select **More**, then **Sources** or **Destinations**.

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:

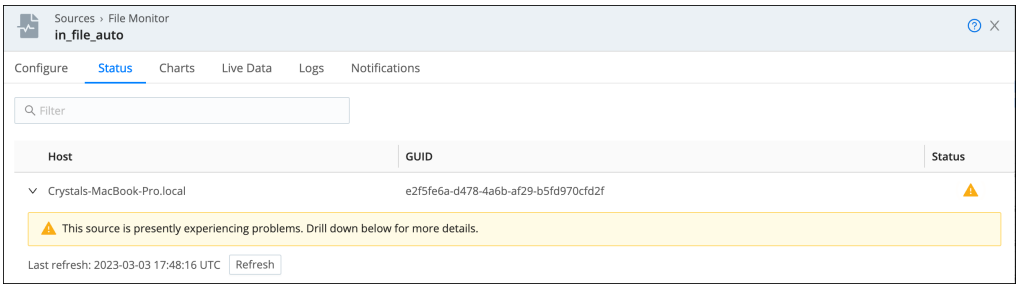
1. Select a Fleet to open it.
2. Select the Fleet Settings tab.
3. Navigate to **General Settings** > **Limits** > **Metrics**.

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.

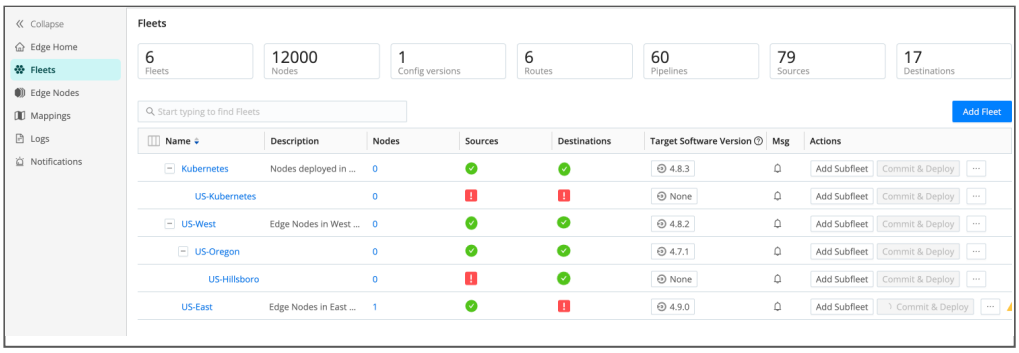


Health status details

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.

Monitor Fleet Information

To view monitoring information for your Fleets, select **Fleets** from the sidebar.

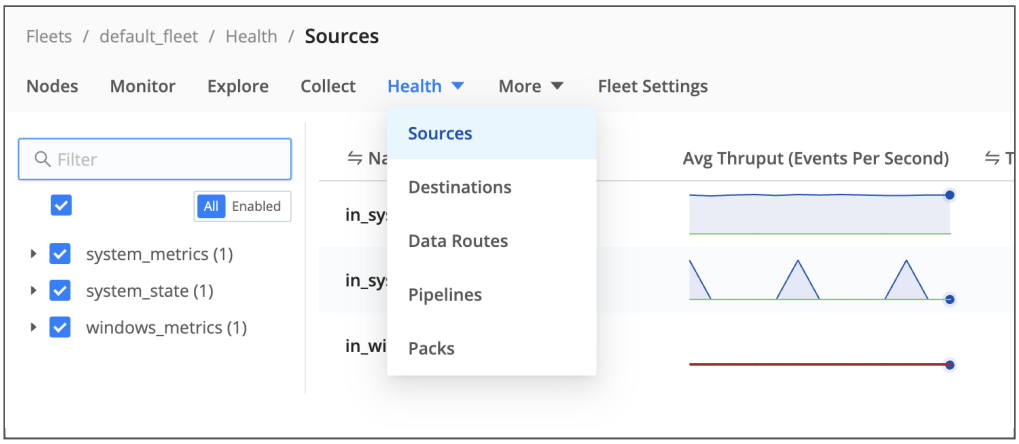


The **Fleets** page gives you access to more information about each of your Fleets and the health of Sources and Destinations associated with the Fleets.

You can open an individual **Fleet** using its link, or use the **Search** field.

View Health Statuses for Resources in a Fleet

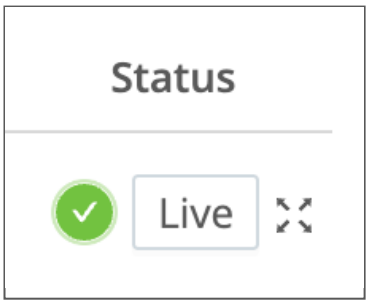
Within any individual Fleet, you can use the **Health** drop-down to view health data for Sources, Destinations, Data Routes, Pipelines, and Packs.



Fleet Health drop-down listing available resource options

In this view, Cribl Edge condenses busy displays to sparklines for legibility.

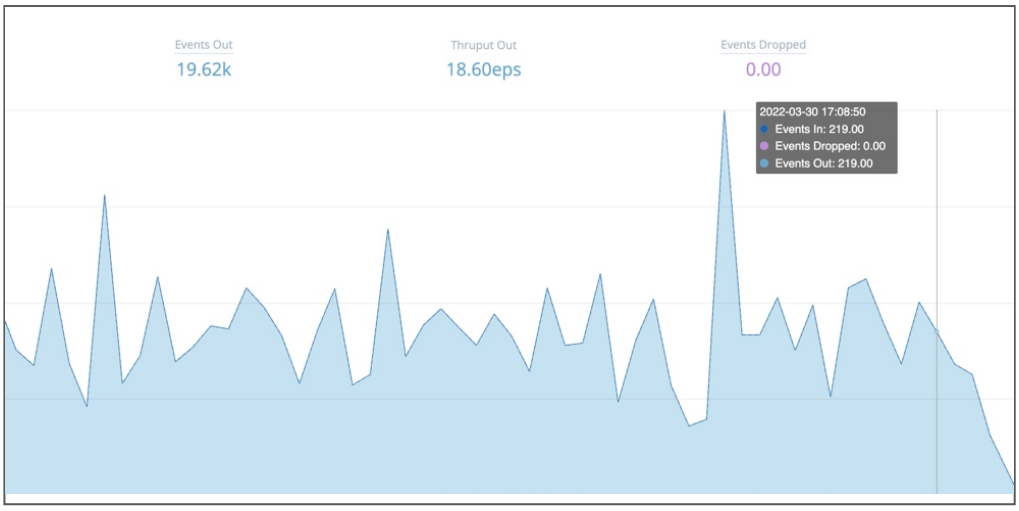
The **Status** column displays the health of each resource with a colored icon. Select the icon to see more details. If a Fleet's **Health** tab is empty, health status might not be enabled because only minimal metrics are being sent. See Enable Health Status for remediation instructions.



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.



Throughput details

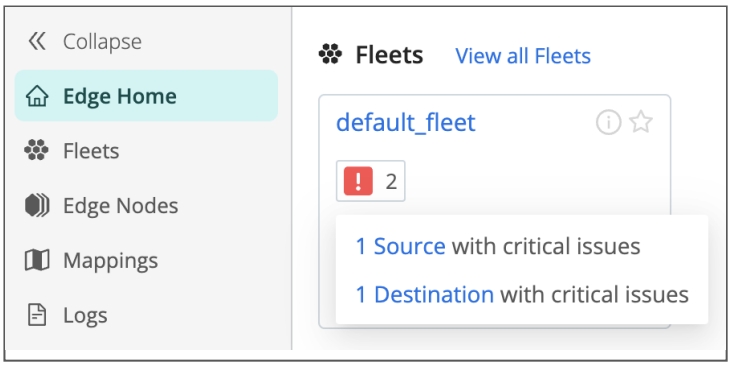
Fleet-level Source and Destination Health Status

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.



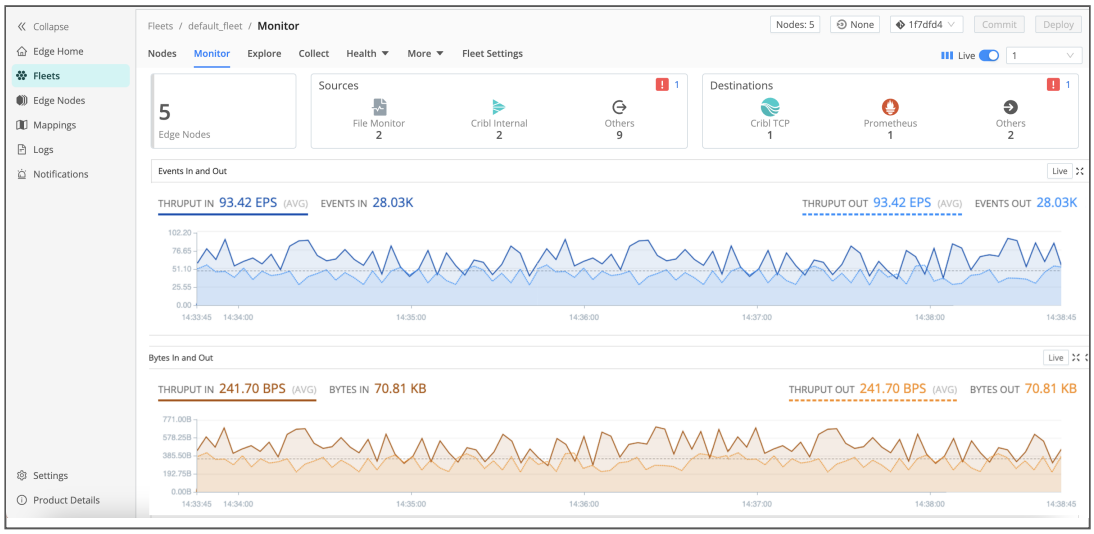
Fleet-level Source and Destination health status icons

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.

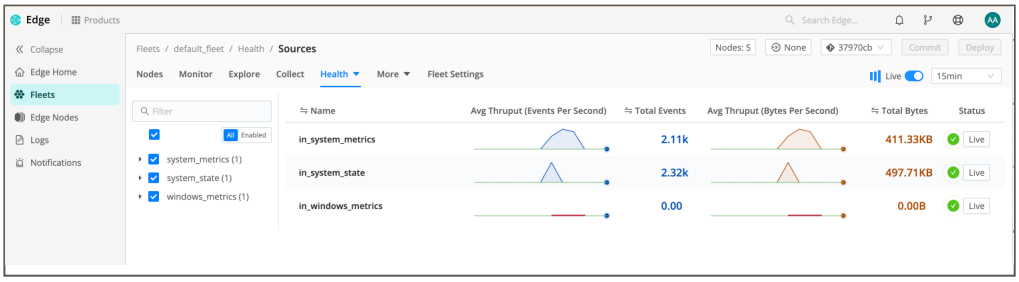


The Monitor tab of a selected Fleet

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

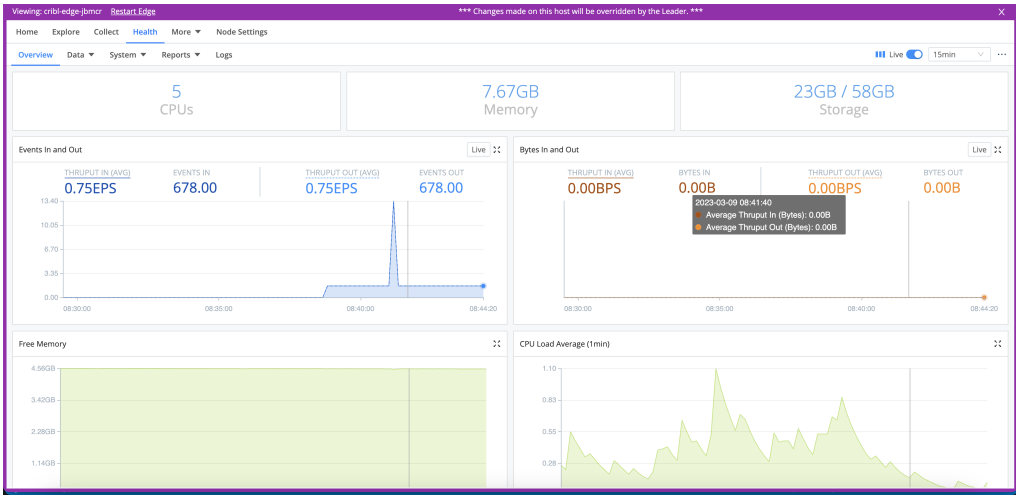
You can review the health status of an Edge Node in two ways: teleport into the Node, or use the Node Info drawer.

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:

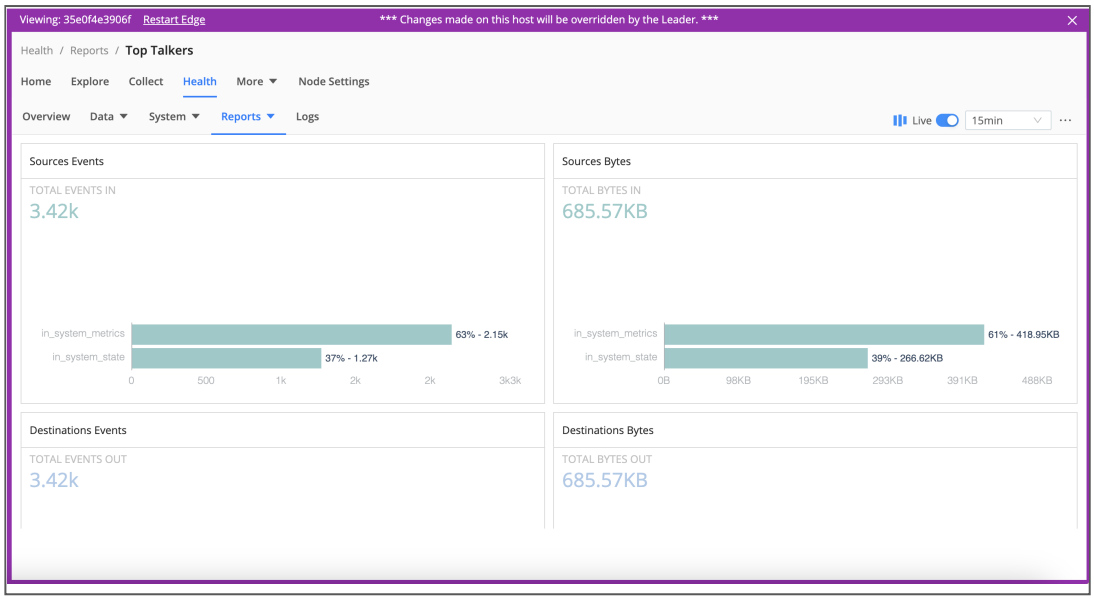
- Queues (Sources)
- Queues (Destinations)

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

Query this endpoint on any instance to check the instance's health. For details, see Querying the Health Endpoint.

Types of Logs

Cribl Edge provides the following log types, by originating process:

- API Server Logs – These logs are emitted primarily by the `API/main` process. They correspond to the top-level `cribl.log` that shows up on the Diag page. These include telemetry/license-validation logs. Filesystem location: `$CRIBL_HOME/log/cribl.log`
- Edge Node Process(es) Logs – These logs are emitted by all the Edge Node Processes, and are very common on single-instance deployments and Edge Nodes. Filesystem location:

`$CRIBL_HOME/log/worker/N/cribl.log`

- Fleet Logs – These logs are emitted by all processes that help a Leader Node configure Fleets. Filesystem location: `$CRIBL_HOME/log/group/GROUPNAME/cribl.log`

For details about generated log files, see Internal Logs. To work with logs in Cribl Edge's UI, see Search Internal Logs.

Log Rotation and Retention

Cribl Edge rotates logs every 5 MB, keeping the most recent 5 logs. Logs will reach the 5 MB threshold more quickly with verbose logging settings (such as `debug`) and with high volumes of system activity. In a Distributed deployment (Edge), all Edge Nodes forward their metrics to the Leader Node, which then consolidates them to provide a deployment-wide view.

For long-term retention, Cribl recommends sending logs from the Leader, and the Cribl Internal > CriblLogs Sources on Edge Nodes, to a downstream service of your choice. See the next section for details.

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.

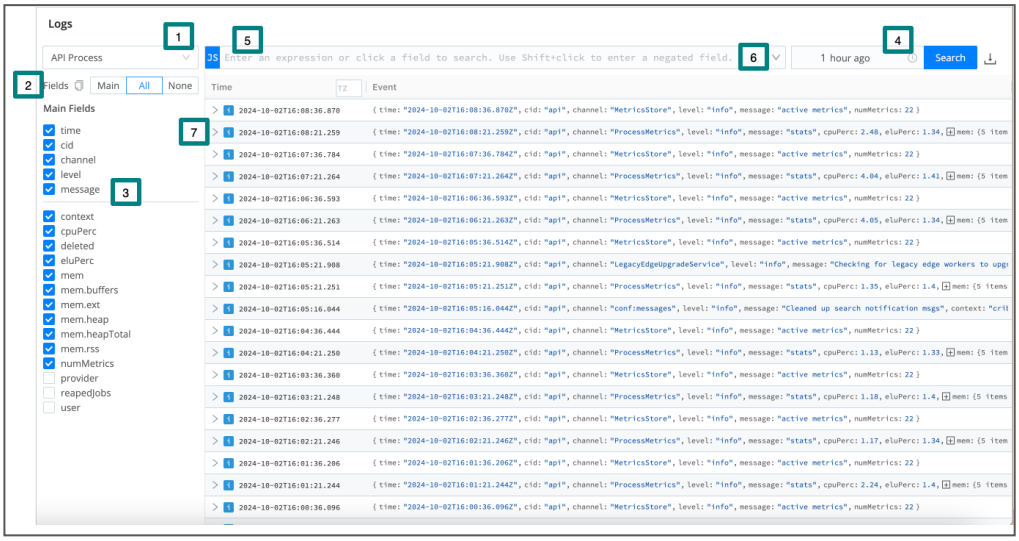
CriblMetrics Override

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.

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 to display. (The **Custom Time Range** pickers use 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.
6	Search box history	Select the history arrow to retrieve recent queries.

Callout	Field	Description
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
`error`	Logs only critical issues that have occurred, 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.

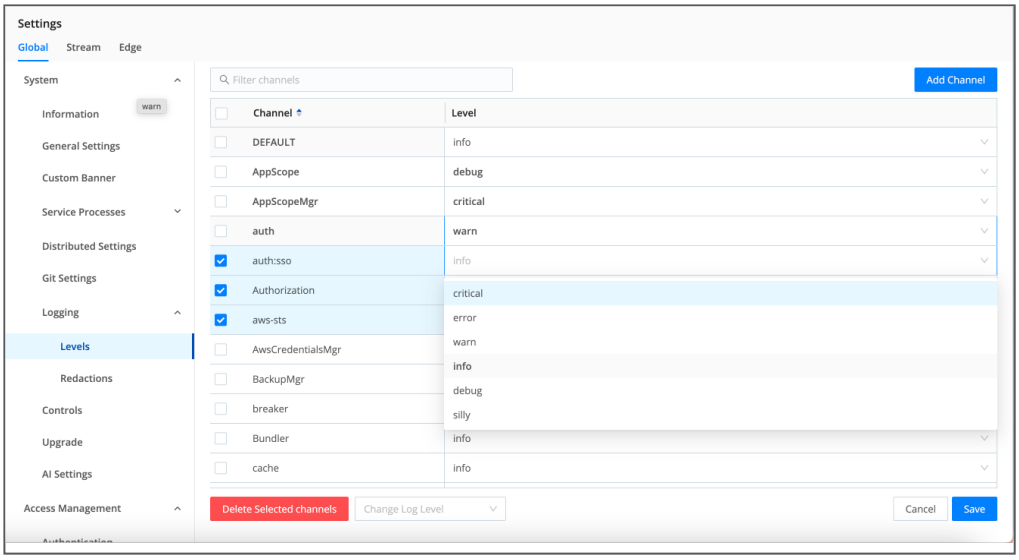
Change Logging Levels

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
Default fields	Cribl Edge always redacts these fields, and you can't modify this list to allow any of them through. However, you can use the two adjacent fields to define stricter redaction.
Additional fields	Type or paste in the names of extra fields you want to redact. Use a tab or hard return to confirm each entry.
Custom redact string	Unless this field is empty, it defines a literal string that will override Cribl Edge's default redaction pattern (explained below) on the affected fields.

Default Redact String

By default, Cribl Edge transforms this page's selected fields by applying the following redaction pattern:

- Echo the field value's first two characters.
- Replace all intermediate characters with a literal `...` ellipsis.
- Echo the value's last two characters.

Anything you enter in the **Custom redact string** field will override this default `??...??` pattern.

Querying the Health Endpoint

Each Cribl Edge instance exposes a `health` endpoint – typically used in conjunction with a Load Balancer – that you can use to make operational decisions.

A healthy response (`200 – Healthy` response code) indicates only that the server is running. It does not mean that all the instance's configured Sources are functioning correctly.

Health Check Endpoint	Healthy Response
`curl http(s)://<host>:<port>/api/v1/health`	`{"status":"healthy","startTime":1617814717110}`

Specifically, the `health` endpoint can return one of the following response codes:

- 200 – Healthy.
- 420 – Shutting down.
- 503 – HTTP engine reports server busy: too many concurrent connections (configurable).

In the above curl example, `<port>` stands for the API port (by default, `9420`).

Several HTTP-based Sources provide the option to enable a Source-level health check endpoint. These use `http(s)://<host>:<port>/cribl_health` and you can find the option under the **Advanced Settings** tab for each Source.

11.1. 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 the Cribl Internal Source in the Fleet, then enable **CriblMetrics**.

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

Five specific metrics are most valuable for monitoring system health. The first two are Cribl Edge composite metrics; the remaining three report on your hardware or VM infrastructure. Because 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.

- `health.inputs`
- `health.outputs` – see the JSON Examples below for both `health.` 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.)

All of the above metrics take these three 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 }
  ],
```

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.

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:

- `cribl.logstream.total.activeCxn` – Total active inbound TCP connections.
- `cribl.logstream.pipe.in_events` – Inbound events per Pipeline.
- `cribl.logstream.pipe.out_events` – Outbound events per Pipeline.
- `cribl.logstream.pipe.dropped_events` – Dropped events per Pipeline.
- `cribl.logstream.metrics_pool.num_metrics` – The total number of unique metrics that have been allocated into memory.
- `cribl.logstream.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.
- `cribl.logstream.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).
- `cribl.logstream.backpressure.outputs` – Destinations experiencing backpressure, causing events to be either blocked or dropped.
- `cribl.logstream.blocked.outputs` – Blocked Destinations. (This metric is more restrictive than the one listed just above.)
- `cribl.logstream.pq.queue_size` – Current queue size, in bytes, per Worker Process.
- `cribl.logstream.host.in_bytes` – Inbound bytes from a given host (host is a characteristic of the data).
- `cribl.logstream.host.in_events` – Inbound events from a given host (host is a characteristic of the data).

- `cribl.logstream.host.out_bytes` – Outbound bytes from a given host (host is a characteristic of the data).
- `cribl.logstream.host.out_events` – Outbound events from a given host (host is a characteristic of the data).
- `cribl.logstream.index.in_bytes` – Inbound bytes per index.
- `cribl.logstream.index.in_events` – Inbound events per index.
- `cribl.logstream.index.out_bytes` – Outbound bytes per index.
- `cribl.logstream.index.out_events` – Outbound events per index.
- `cribl.logstream.route.in_bytes` – Inbound bytes per Route.
- `cribl.logstream.route.in_events` – Inbound events per Route.
- `cribl.logstream.route.out_bytes` – Outbound bytes per Route.
- `cribl.logstream.route.out_events` – Outbound events per Route.
- `cribl.logstream.source.in_bytes` – Inbound bytes per Source.
- `cribl.logstream.source.in_events` – Inbound events per Source.
- `cribl.logstream.source.out_bytes` – Outbound bytes per Source.
- `cribl.logstream.source.out_events` – Outbound events per Source.
- `cribl.logstream.sourcetype.in_bytes` – Inbound bytes per sourcetype.
- `cribl.logstream.sourcetype.in_events` – Inbound events per sourcetype.
- `cribl.logstream.sourcetype.out_bytes` – Outbound bytes per sourcetype.
- `cribl.logstream.sourcetype.out_events` – Outbound events per sourcetype.
- `nodecluster.primary.messages` – Number of IPC (interprocess communication) calls made by a Worker/Edge Node Process to the API Process.

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`.
- `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.
- `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.

The `ci` and `co` (duplicate) dimensions are deprecated as of Cribl Edge 4.1.2, and will be removed in a future version. Where any of your code relies on these dimensions, Cribl recommends that you promptly substitute their respective replacements, `input` and `output`. This applies to relevant Cribl Functions and Pipelines, dashboards in downstream services, etc.

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:

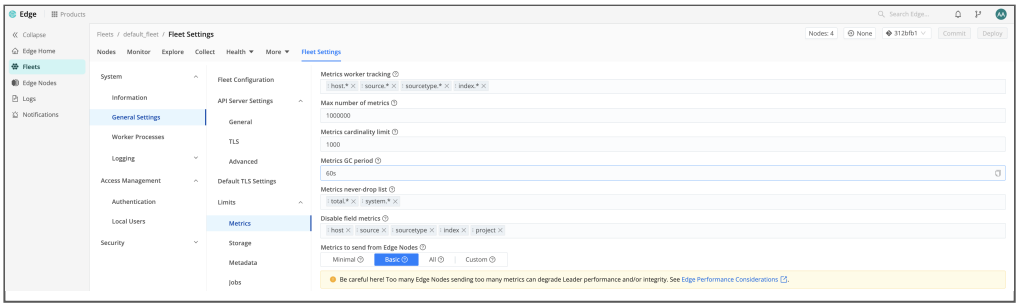
- Single-instance: **Settings** > **System** > **General Settings** > **Limits** > **Metrics**.
- Distributed Leader: **Settings** > **Global Settings** > **System** > **General Settings** > **Limits** > **Metrics**.
- Distributed Fleets: [Select a Fleet] > **Fleet Settings** > **System** > **General 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 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 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:
		
		`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,

Set	Purpose	List of Metrics Sent
		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. Note that 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

The `cpus` metric returns an array of CPU/memory key-value pairs. This provides an alternative way to understand CPU utilization, but it requires you to query all your CPUs individually.

11.2. 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.

Currently, most logs listed on this page are available only in on-prem deployments. Cribl.Cloud currently supports only Edge Node logs, and only on customer-managed hybrid 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, e.g., `GET /api/v1/version/info`.	**Leader** > **Access**
`audit.log`	Actions pertaining to files, e.g., `create`, `update`, `commit`, `deploy`, `delete`.	**Leader** > **Audit**
`notifications.log`	Messages that appear in the Notification list in the UI.	**Leader** > **Notifications**
`ui-access.log`	Interactions with different UI components described as URLs, e.g., `/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, etc. 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, etc.	**GROUPNAME** > **Config helper**

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 (i.e., with its API Process) and other API requests, e.g., 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**

Logfile Name	Description	Equivalent on Logs page
`metrics.log`	Messages about the Worker/Edge Node's outbound HTTP request statistics.	**GROUPNAME** > **Worker:HOSTNAME** > **Worker Process N**

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**

The Edge Node 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","mes
```


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; e.g., 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.

Field	Description
`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`.
`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.

Troubleshooting With the `cribl_stderr` Log

In rare instances, Cribl Edge's Node.js backend may encounter an OOM or fatal error, preventing logging to its usual location. To aid in troubleshooting, a special `cribl_stderr.log` file is created, containing timestamped error details in UTC. This file is intended for Cribl Support and is not accessible through the UI.

To prevent excessive disk usage, Cribl Stream implements log rotation for `cribl_stderr.log`, limiting files to 5MB each with a maximum of 5 rotated copies. This addresses an issue where the log file could rapidly expand

due to logged errors, impacting system performance.

11.3. Tracking Data Activity For An Edge Node

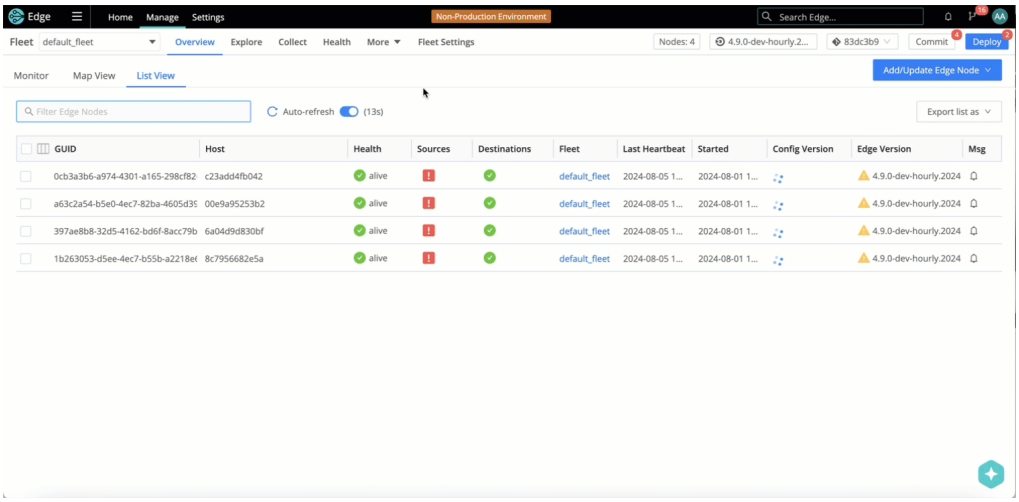
Monitoring and troubleshooting your Edge Nodes guarantees the integrity and reliability of the data you're collecting from your Nodes.

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.

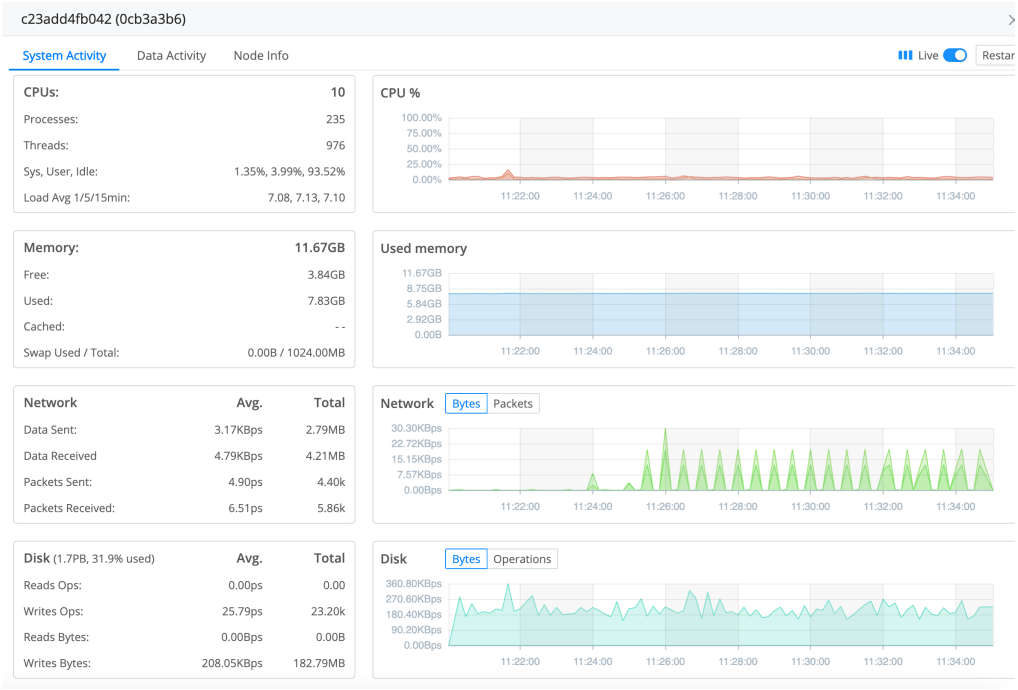


Node info panel

With **Teleport 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.

Viewing the System Activity for a Node

To view system activity, click the row for an Edge Node.



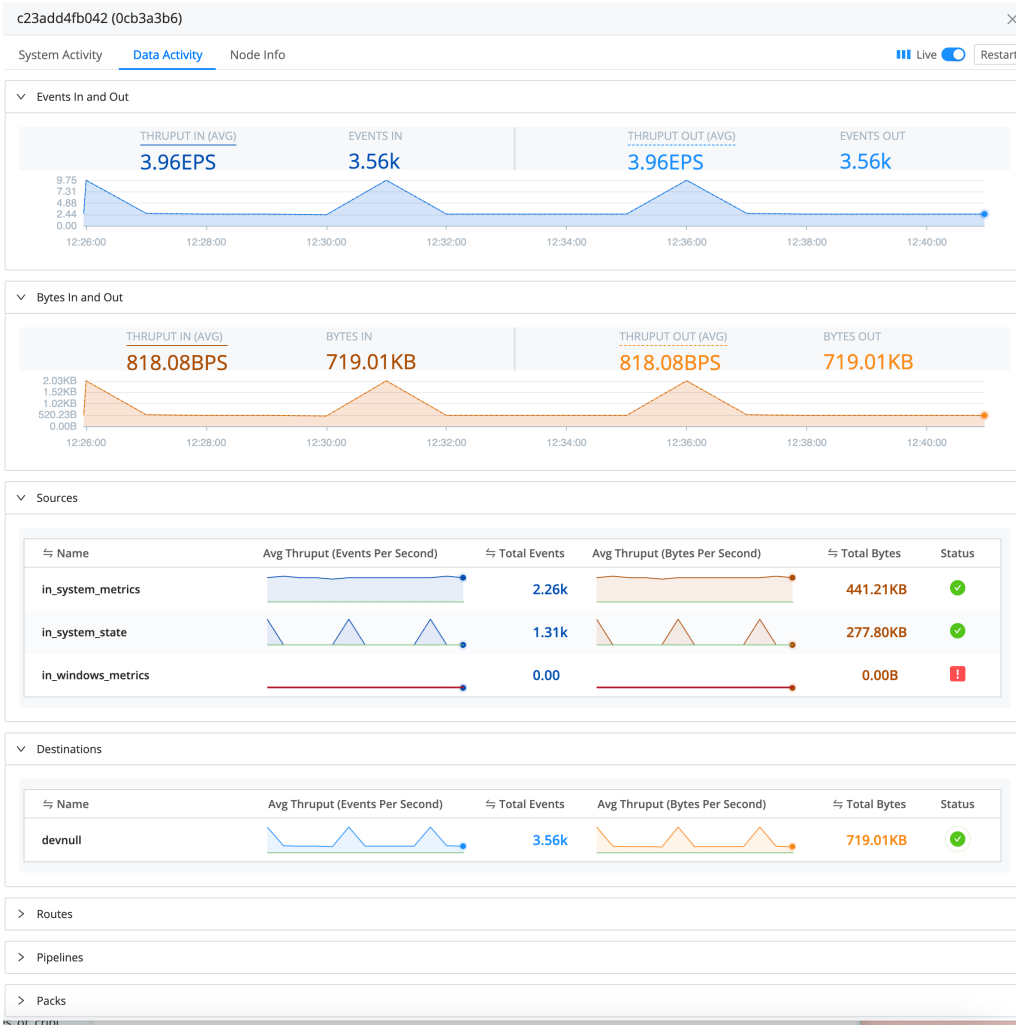
Node info panel system activity

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.
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.

Viewing 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:

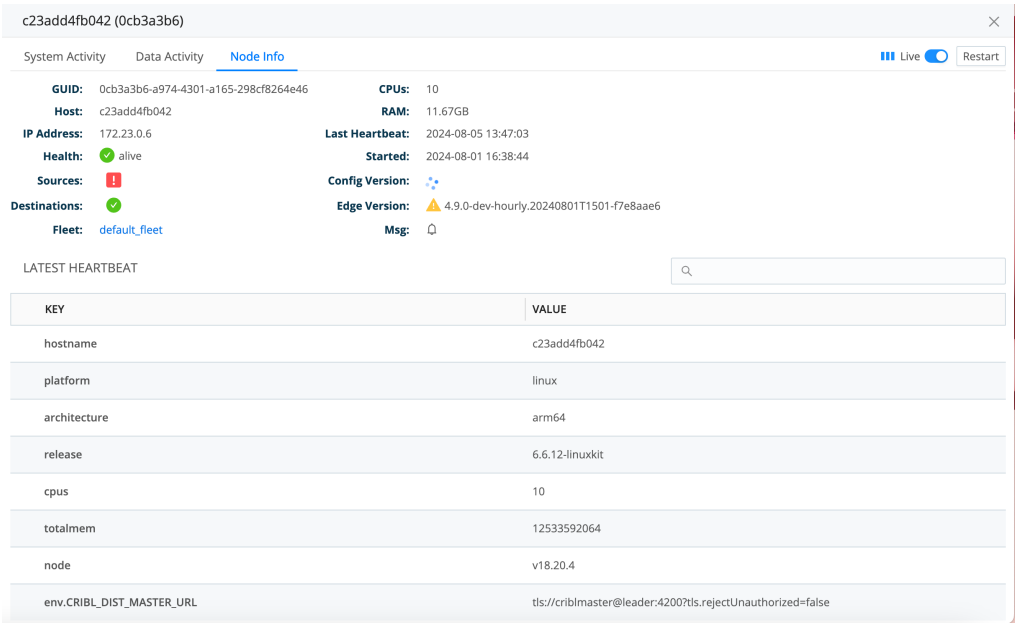
Field	Description
	• The Destination name, average throughput (events per second), total events, average byte throughput (bytes per second), and total bytes. • The Destination health status, which is useful for identifying Nodes experiencing Destination issues and for diagnosing issues with Destinations 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.

Viewing Additional Edge Node Info

To view detailed information, click the row for an Edge Node.



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 • Date started • Source health • Destination health • Fleet • Config version • Edge version • Messages .
Key:Value pairs	When an Edge Node first connects to the Leader, it sends an initial report, which is the data shown here. This report includes specifics about the operating system, the underlying infrastructure (such as Kubernetes or EC2), and the Cribl configuration settings.

12. Working With Data

12.1. Event Model

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 epoch format>",
  "__inputId": "<Id/Name of Source that delivered the event>",
  "__other1": "<Internal field1>",
  "__other2": "<Internal field2>",
  "__otherN": "<Internal fieldN>",
  "key1": "<value1>",
  "key2": "<value2>",
  "keyN": "<valueN>",
  "...": "..."
}
```

Some notes about these representative fields:

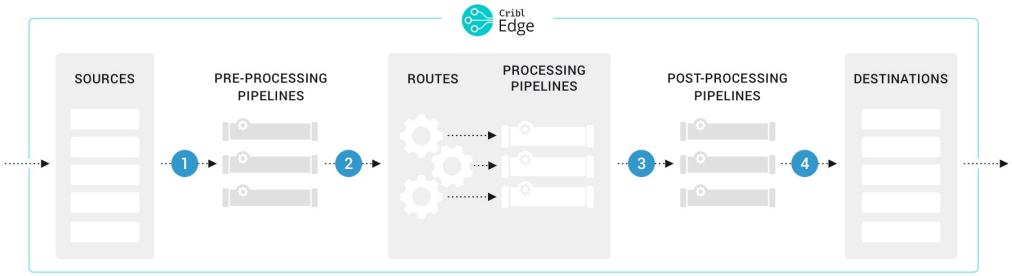
- Fields that start with a double-underscore are known as **internal fields**, and each Source can add one or many to each event. For example, Syslog adds both a `__inputId` and a `__srcIpPort` field. Internal fields are used only within Cribl Edge, so they are serialized only to Cribl internal Destinations.
- 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 epoch format) 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 Preview (right pane):

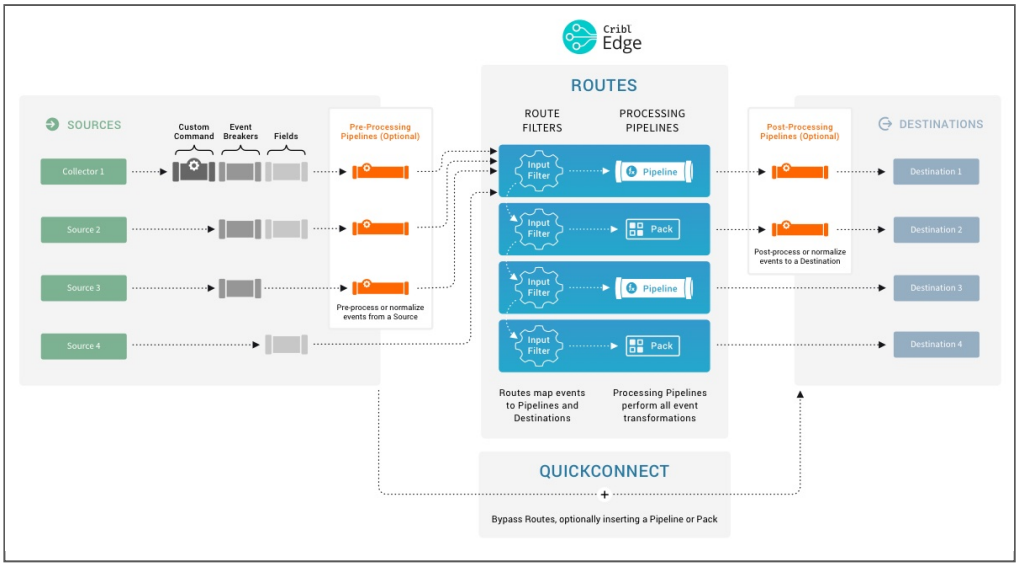
1. Click **Start a Capture**.
2. In the resulting modal, enter a **Filter expression** to narrow down the events of interest.
3. Click **Capture...** and (optionally) change the default Time and/or Event limits.
4. Select the desired **Where to capture** option. There are four options:

- **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.



12.2. Event Processing Order

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, 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. Auth tokens are applied at this point on Splunk HEC Sources. (Details here.)
5. 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.
6. Auth tokens are applied at this point on HTTP-based Sources other than Splunk HEC (e.g., Elasticsearch API Sources).
7. Pre-processing Pipeline: Optionally, you can use a single Pipeline to condition (normalize) data from this input before the data reaches the Routes.
8. 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.

9. 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.
10. Post-processing Pipeline: Optionally, you can append a Pipeline to condition (normalize) data from each Processing Pipeline before the data reaches its Destination.
11. 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.)

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.

12.3. 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

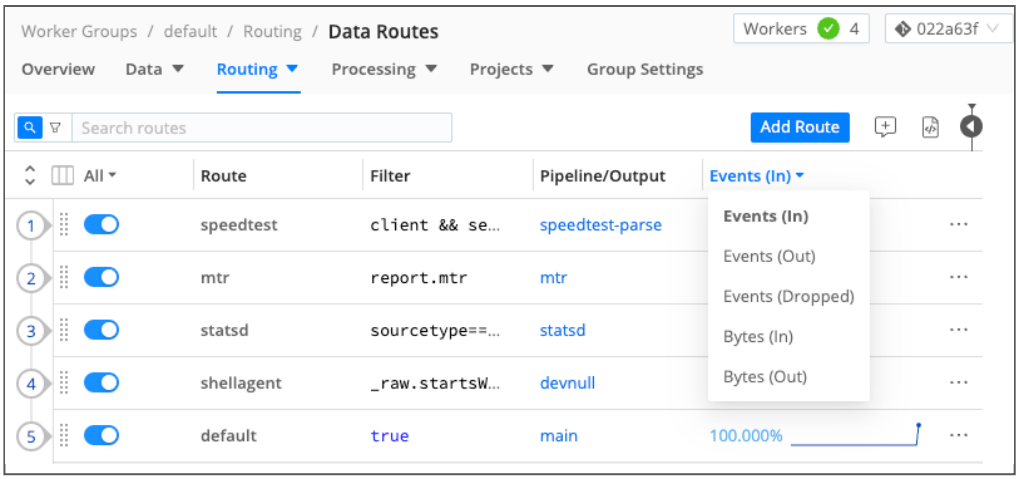
Routes apply filter expressions on incoming events to send matching results to the appropriate Pipeline. Filters are JavaScript-syntax–compatible expressions that are configured with each Route. Examples are:

- `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. (Sources and Destinations can also have their own attached pre-/post-processing Pipelines.)

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, top->down. The stats shown in the **Bytes/Events** (toggle) column are for the most-recent 15 minutes.



Routes and bytes

In the example above, incoming events will be 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 for any event that does not match any of the other Routes.

Above, note the **Events In** drop-down menu to toggle between displaying Events versus Bytes, and to display In, Out, or Dropped.

Displayed percentages of Events and 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.

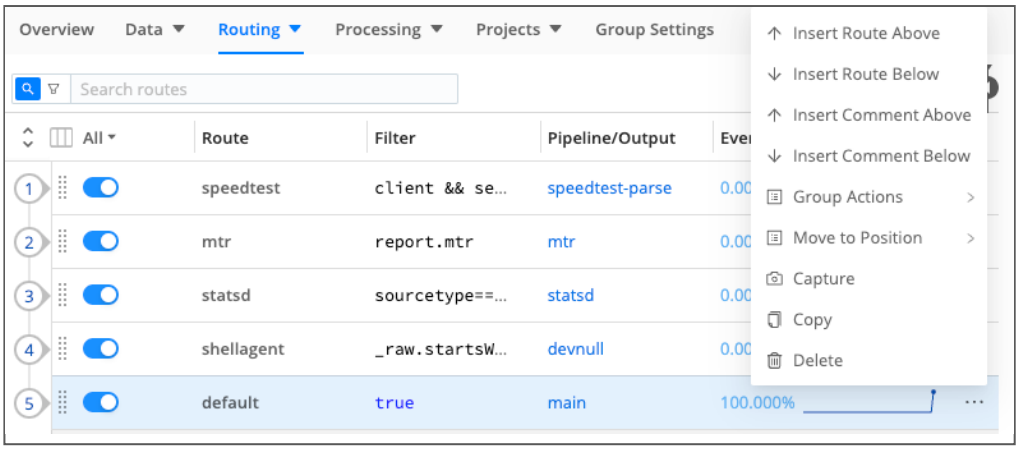
Managing the Routes Page

To apply a Route before another, simply drag it vertically. Use the toggles to turn Routes `On/Off` inline, as necessary, to facilitate development and debugging.

You can press the `]` (right-bracket) shortcut key to toggle between the 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:

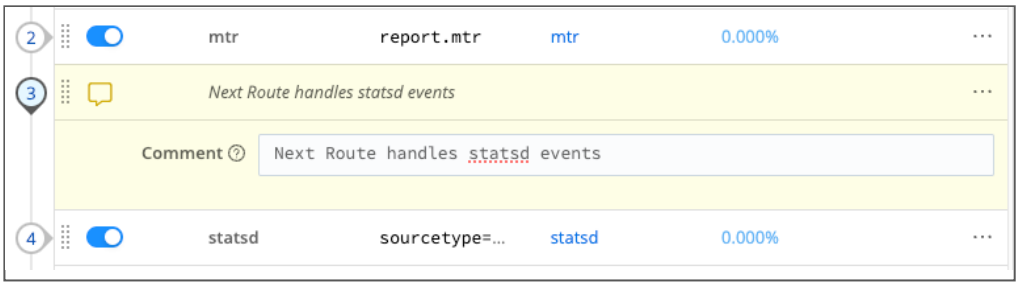
- Insert, group, move, copy, or delete Routes.
- Insert comments above or below Routes.
- Capture sample data through a selected Route.



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.

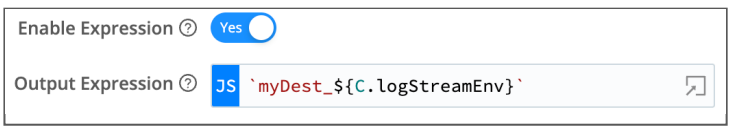


Comments make the Routing table self-documenting

Output Destination

You can configure each Route with an **Output** that defines a Destination to send events to, after they're processed by the Pipeline.

If you toggle **Enable expression** to `Yes`, the **Output** field changes to an **Output 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.)



Output expression enabled

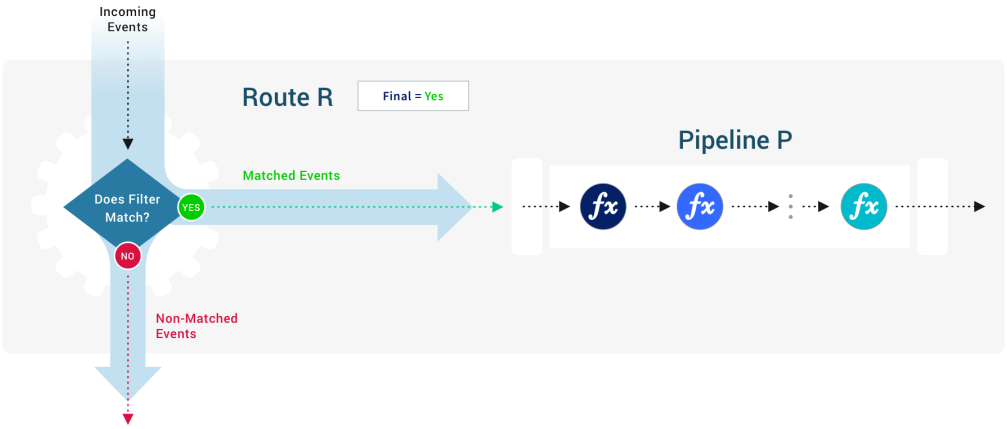
Expressions that match no Destination name will silently fail.

> On Routes within Packs, neither **Output** control is available, because Packs cannot specify Destinations.

The `Final` Toggle

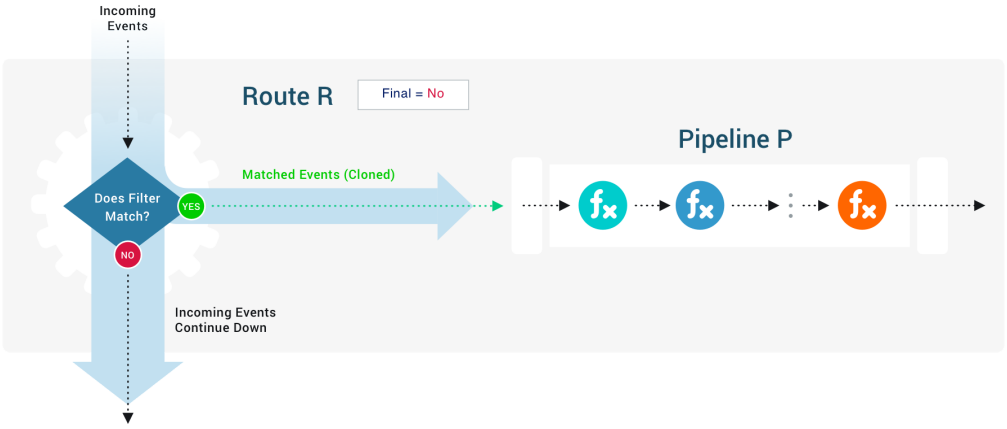
The `Final` toggle in Route settings controls what happens to an event after a Route-Pipeline pair matches it.

When `Final` is toggled to `Yes` (default), the Route "consumes" an event, meaning the event will not pass down to any other Route below.



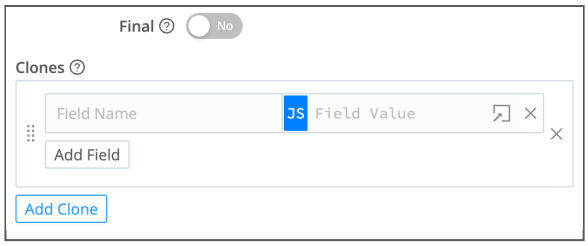
The Route is Final and non-matched events stop here

When `Final` is toggled to `No`, the event will be processed by this Route, **and** then passed on to the next Route below.



The Route isn't Final and all events pass further on

When you toggle `Final` to `No`, 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 to `Yes` 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.

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 the `Final` flag enabled, 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 **Output** of a Route higher in the table, duplicate events will reach that Destination.

You can correct this either by setting a Route to `Final`, 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.

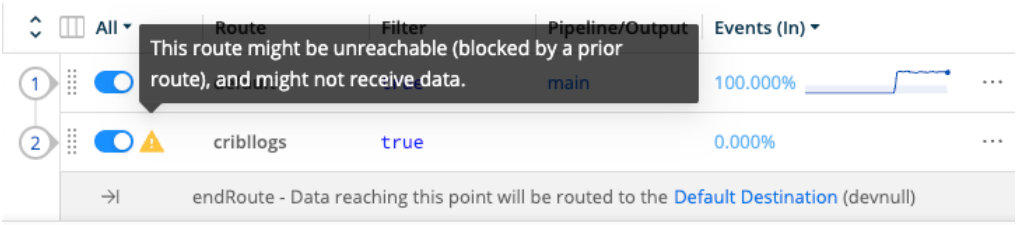
Route Groups

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 is final (**Final** toggle is set to `Yes`).
- Previous Route's **Filter** expression evaluates to true, (e.g., `true`, `1 === 1`, etc.).

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).

Routing with 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.

12.4. 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.

Access Pipelines

To access Routes:

1. In the sidebar, select **Fleets**, then select a Fleet.
2. On the **Fleets** submenu, select **More**, then **Pipelines**.

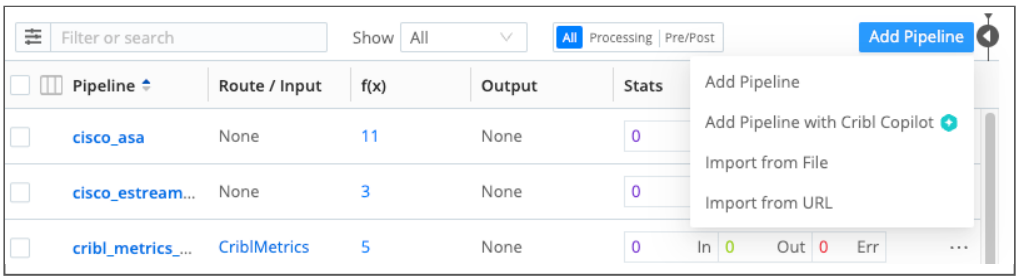
On a Single-instance deployment, select **More** straight from the **Home** submenu.

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 Pipeline assistant to create a Pipeline using AI. See Create Pipelines Using Cribl Copilot 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.)

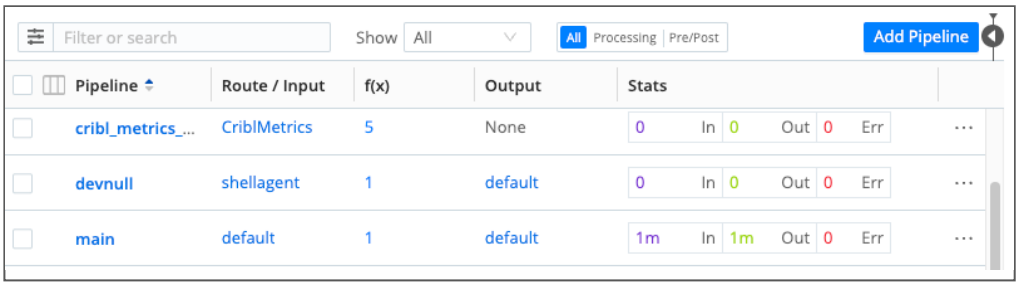


To export a Pipeline, see Advanced Mode (JSON Editor).

To import or export a Pipeline along with broader infrastructure (like Knowledge Objects and/or sample data files), see Packs.

How Pipelines Work

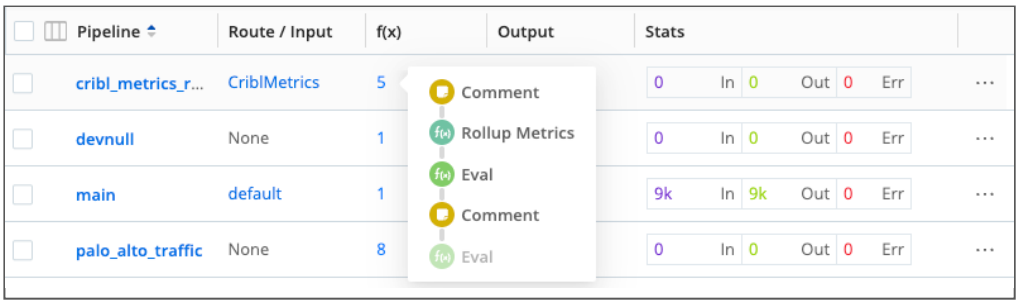
Events are always delivered to the beginning of a Pipeline via a Route. The data in the **Stats** column shown below are for the last 15 minutes.



Pipelines and Route inputs

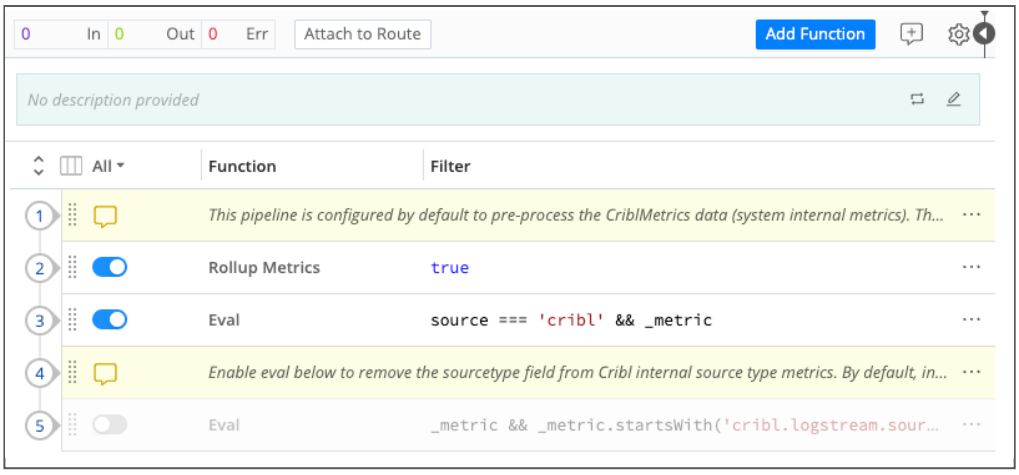
You can press the] (right-bracket) shortcut key to toggle between the Preview pane and an expanded Pipelines display. (This shortcut works when no field has focus.)

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

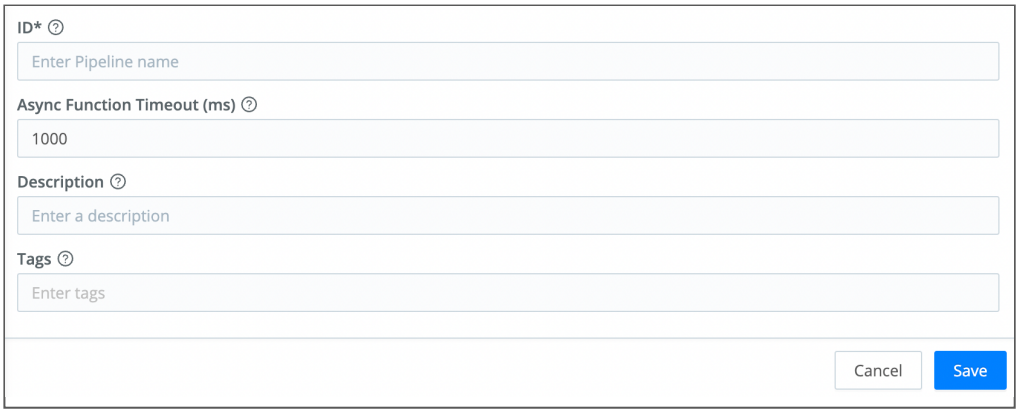
Use the **Attach to Route** link at the top to associate a new Pipeline with a Route.

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

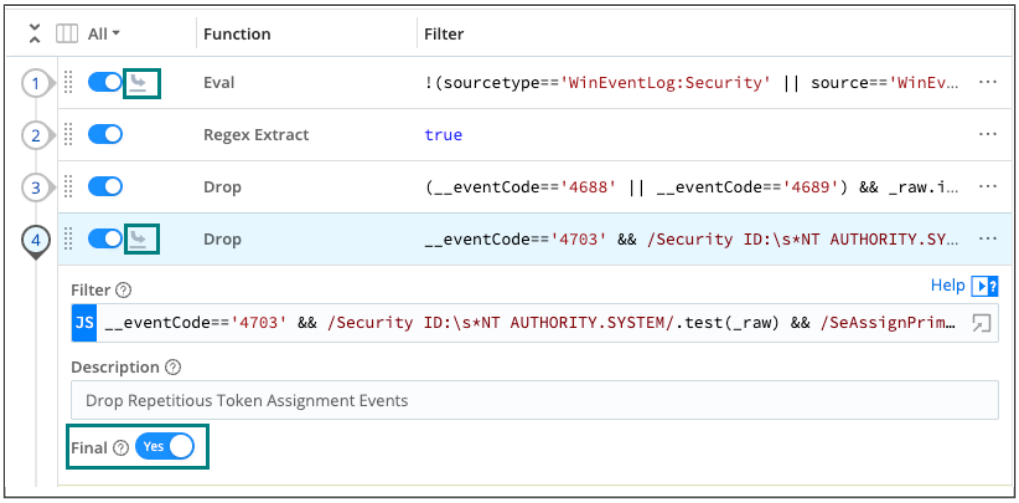
The `Final` Toggle

The `Final` toggle in Function settings controls what happens to the results of a Function.

When `Final` is toggled to `No` (default), events will be processed by this Function, **and** then passed on to the next Function below.

When `Final` is toggled to `Yes`, the Function "consumes" the results, meaning they will not pass down to any other Function below.

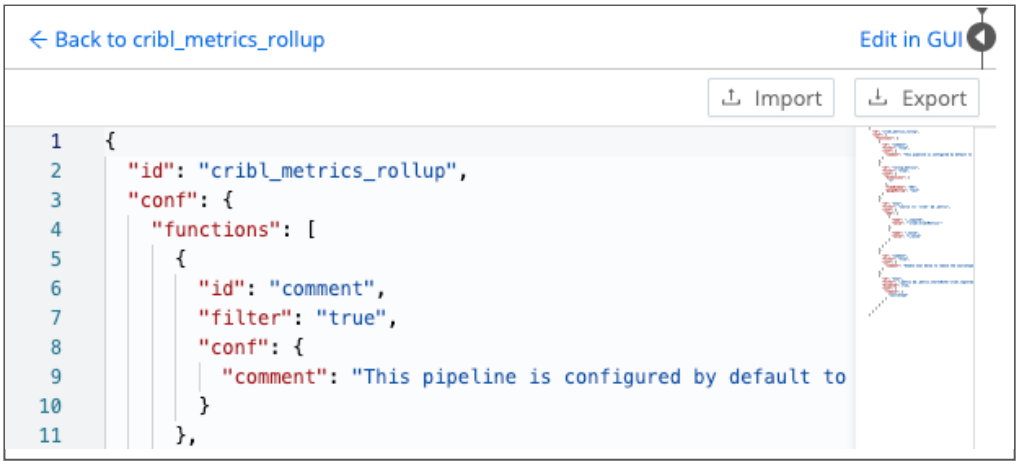
An icon in Pipeline view indicates that a Function is `Final`.



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.



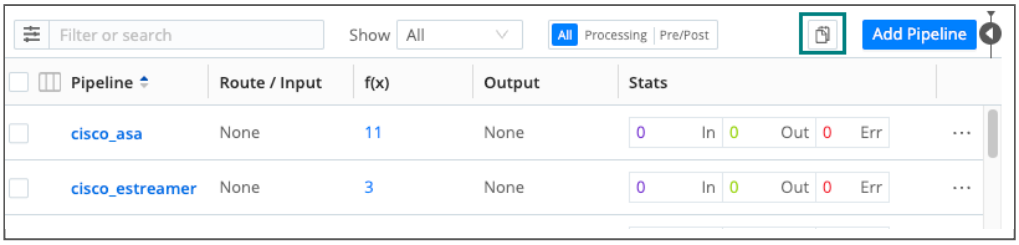
Advanced Pipeline Editing

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

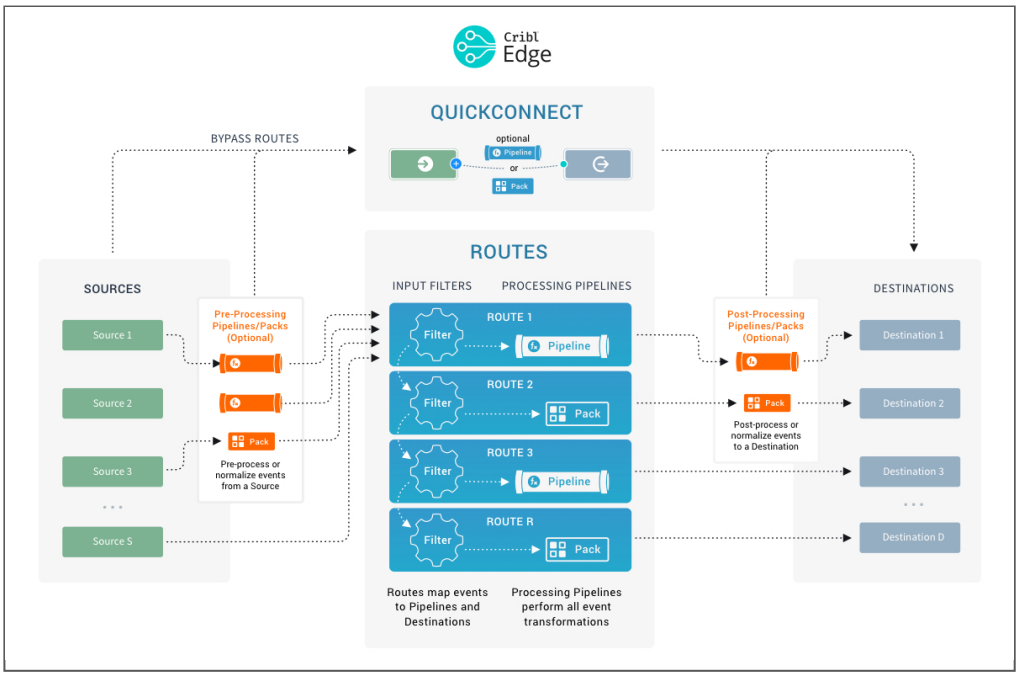
Pasting prompts you to confirm, or change, a modified name for the new Pipeline. The result will be an exact duplicate of the original Pipeline in all but name.

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 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.

As with Routes, the general goal is to minimize extra work that a Function will do. The fewer events a Function has to operate on, the better the overall performance.

For example, if a Pipeline has two Functions, **f1** and **f2**, and if **f1** operates on `source 'foo'` and **f2** operates on `source 'bar'`, it might make sense to apply `source=='foo'` versus `source=='bar'` filters on these two Functions, respectively.

12.4.1. Create Pipelines Using Cribl Copilot

The Cribl Copilot Pipeline assistant in Cribl Stream uses AI to help you create and edit data transformation Pipelines. Simply describe what you want the Pipeline to do using natural language, and Cribl Copilot will create the Pipeline for you. You'll usually get results in about 10 seconds.

The Cribl Copilot Pipeline assistant is available in Cribl.Cloud, and in on-prem installations of Cribl Stream v4.8 and newer. Cribl Copilot always requires internet access.

The Cribl Copilot Pipeline assistant can generate Pipelines of any length using the following Functions:

- CEF Serlializer
- Code
- Comment
- DNS Lookup
- Drop
- Eval
- Flatten
- Mask
- Numerify
- Regex Extract
- Regex Filter
- Rename
- Serialize
- Suppress

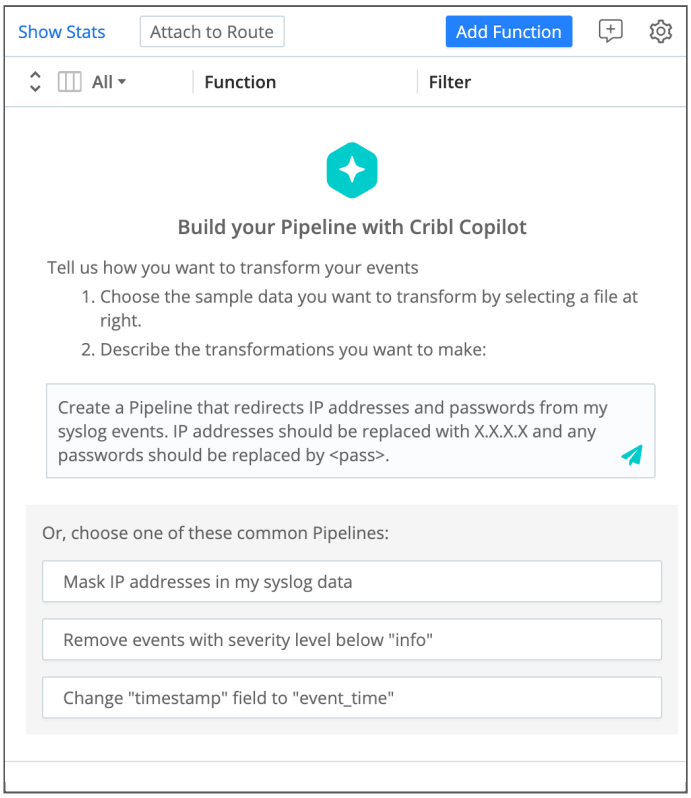
Cribl does not make any warranty regarding the availability, uptime, performance, or accuracy of Cribl Copilot, including PII Detection and Prevention. Please read the Supplemental Terms for Cribl Copilot for more information.

To create a Pipeline:

1. In a Cribl Stream Worker Group, on the Worker Group top menu, select **Processing** and select **Pipelines**.
2. Select **Add Pipeline**, and select **Add Pipeline with Cribl Copilot**. Enter a name for your Pipeline in the **ID** field. The ID can contain only letters (A-Z, a-z), numbers (0-9), and the characters "_" and "-".

Optionally, change the async function timeout, and add a description or tags.
Select **Save**.

3. On the right, select an input event that your Pipeline will apply transformations to.
4. On the left, in the text box under **Build your Pipeline with Cribl Copilot**, type a prompt that describes what you want your Pipeline to do.



Sample Pipeline prompt

Make sure you have an input event selected on the right before trying to send a Pipeline creation prompt. You will know an event is selected when it is outlined in blue.

5. Select Send .
6. Confirm that the Pipeline performs the action you expected. If not, you can make edits manually, or cancel and try entering a new Pipeline description.
7. Save the Pipeline.

Example Prompts

The following are sample prompts you can try to better understand how Cribl Copilot uses information about Cribl Stream, Pipelines, your query, and the sample input event, and combines this information to generate a new Pipeline.

- **Redact PII (personally identifiable information).** Try a prompt like, “Redact any log data that may fall under GDPR compliance restriction”

- **Reduce data storage costs by dropping unnecessary events.** Try a prompt like, “Filter out any logs that have a 200 status code or a null ‘error’ field”
- **Enrich incoming data.** Try a prompt like, “Parse any IP addresses from my logs and then perform a DNS lookup on them. Place the IP and DNS hostname from the results in a separate top-level field.”

How to Use Prompts Successfully

To ensure the best results from the Pipeline assistant:

- **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 Cribl Copilot chatbot 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?`

What the Pipeline Assistant Doesn’t Do

You can’t edit an existing Pipeline Function; Cribl Copilot will always create new Functions and append them to the existing Pipelines. Also, this feature, like any AI, is still learning. It may not always work exactly as expected, so be sure to check the generated output events and logic carefully. Please provide feedback to help Cribl Copilot learn!

What Data the Pipeline Assistant Can Access

The Cribl Copilot Pipeline assistant has access to:

- The natural language query, written by the user.
- The sample input event selected by the user.

Cribl Copilot has access to no other user or Organization data.

12.5. Packs

Packs enable Cribl Edge administrators and developers to pack up and share complex configurations and workflows across multiple Worker Groups, or across organizations.

Packs = Portability

With a Cribl Edge deployment of any size, using Packs can simplify and accelerate your work. Packs can also accelerate internal troubleshooting, and can accelerate working with Cribl Support, because they facilitate quickly replicating your Cribl Edge environment.

For example, where a Pipeline configuration references Lookup file(s), Cribl Edge will import the Pipeline only if the Lookups are available in their configured locations. A Pack can consolidate this dependency, making the Pipeline portable across Cribl Edge instances. If you import the Lookups into the Pack, you can develop and test a configuration, and then port it from development to production instances, or readily deploy it to multiple Fleets.

What Is a Pack?

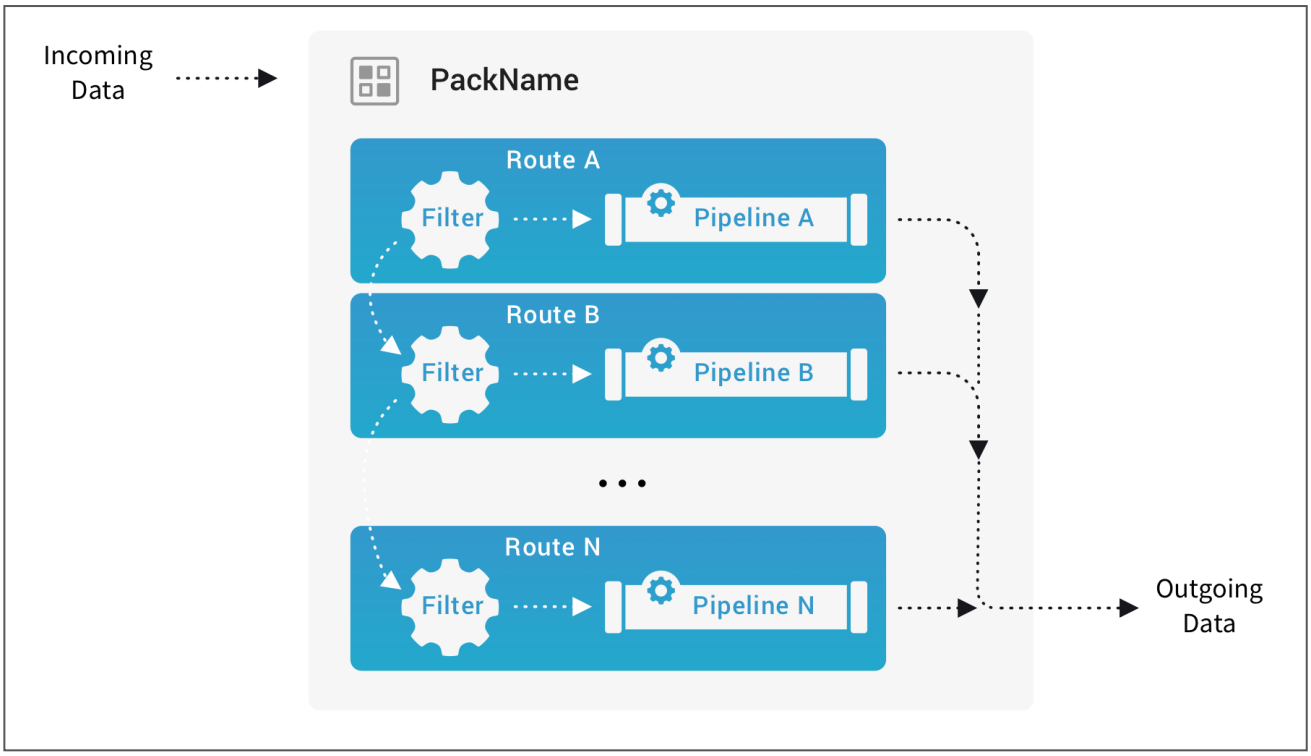
Each Pack provides you with a microcosm of a Cribl Edge deployment. (See the next section for what's included and excluded). A Pack also includes a restricted version of Cribl Edge's user interface (outlined in the first half of the Use Packs section below). You can import and export each Pack as a file, in `.crbl` format.

What's in a Pack?

Currently, a Pack can pack up everything between a Source and a Destination:

- Routes (Pack-level).
- Pipelines (Pack-level).
- Functions (built-in and custom).
- Sample data files.
- Knowledge objects (Lookups, Parsers, Global Variables, Grok Patterns, Schemas, and more).

As the above list suggests, a Pack can encapsulate a whole set of infrastructure for a given use case.



A Pack with internal Routes & Pipelines; no Knowledge or samples

Packs don't have access to Knowledge objects (Lookups, Global Variables, and so forth) that are defined outside the Pack. Knowledge objects defined **within** a Pack are not available to Cribl Edge resources outside the Pack. In either scenario, you must explicitly duplicate the Knowledge objects in the opposite scope.

What's Not in a Pack?

Sources, Collectors, and Destinations are external to Packs, so you can't specify them within a Pack. This excludes a few other things:

- Routes configured within a Pack can't specify a Destination.
- Packs can't include Event Breakers, which are associated with Sources. (However, you can instead use the Event Breaker Function in Pipelines that you include in your Pack.)

You connect a Pack with a Source and Destination by attaching it to a Route, just as you'd attach a Pipeline. For details, ssee Where Can I Use Packs? below.

Where Can I Get Some Packs?

See The Cribl Packs Dispensary below.

Use Packs

These instructions cover using predefined Packs, as well as creating and modifying Pack configurations.

Version Compatibility

Packs created or modified in Cribl Edge 4.0.x cannot be used in any pre-4.0 version of Cribl Edge. (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 later.

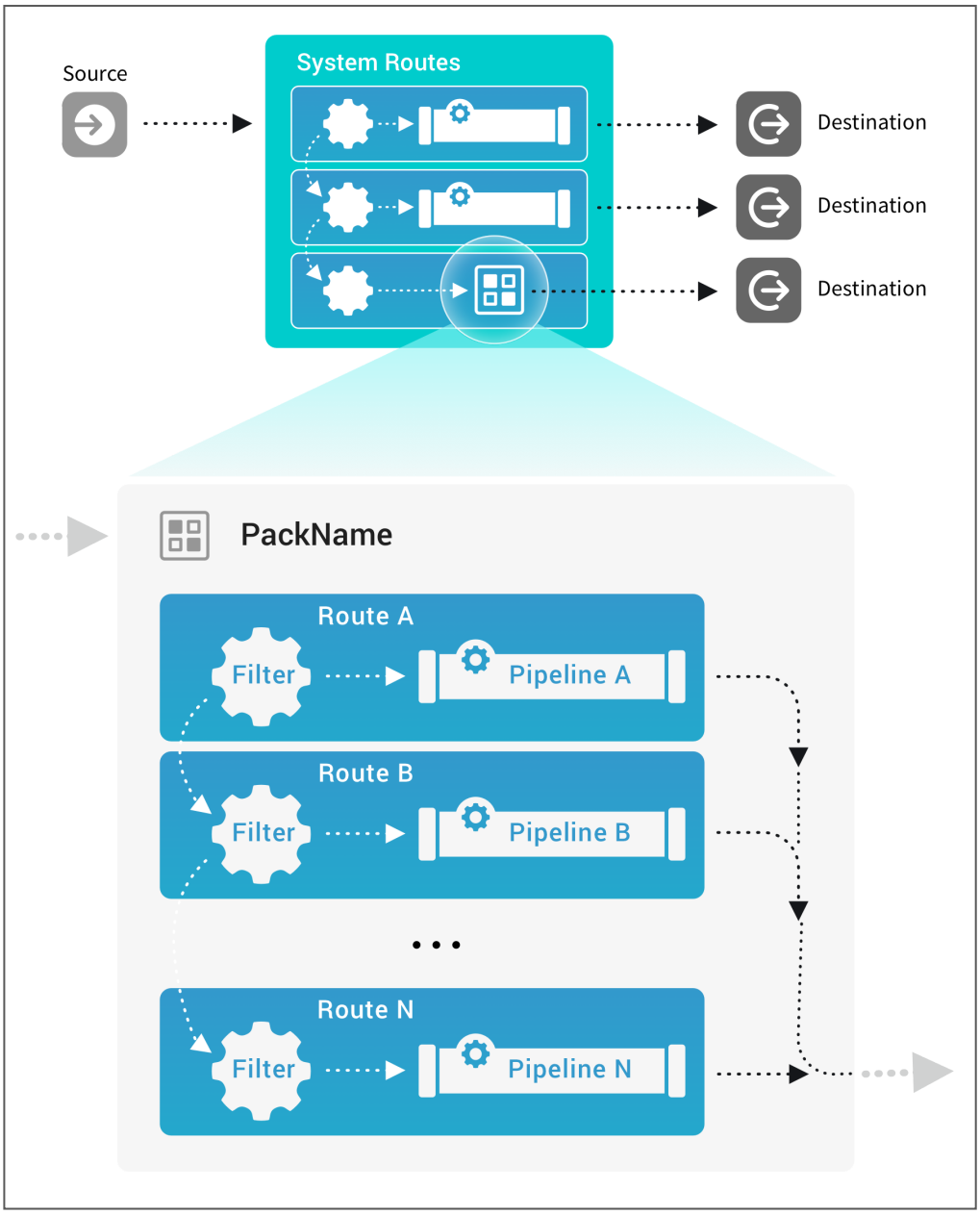
For compatibility questions about any individual Packs, please contact us in the Cribl Community Slack `#packs` channel.

Where Can I Use Packs?

Wherever you can reference a Pipeline, you can specify a Pack:

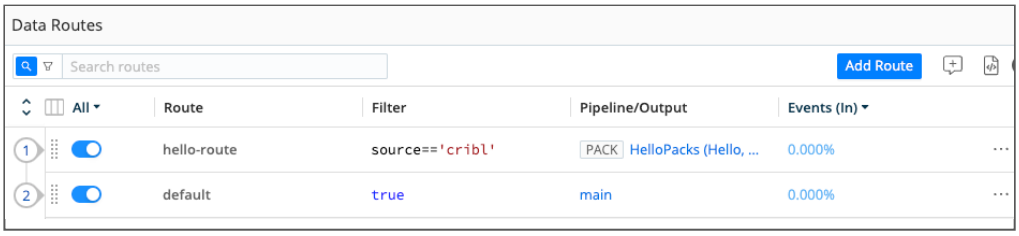
- In Sources, where you attach pre-processing Pipelines.
- In Destinations, where you attach post-processing Pipelines.
- In Routes, in the Routing table's **Pipeline/Output** column.

This expanded view shows how a Pack can replace a Pipeline in a Route:



A Pack snaps into Cribl Edge like an enhanced Pipeline

Packs are distinguished in the display with a **PACK** badge, as you can see here in the Routing table:



PACKs badged 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).

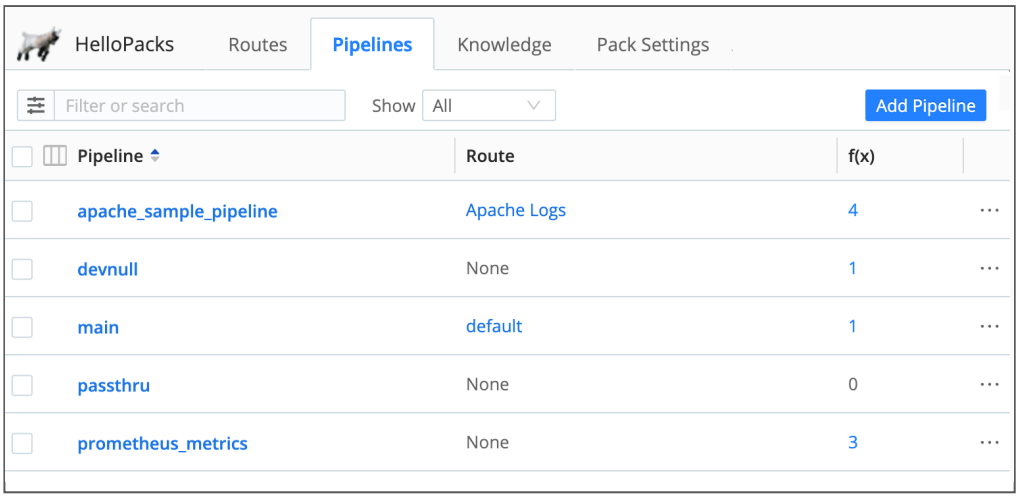
Each Group's Packs are stored at `$CRIBL_HOME/groups/<group-name>/default/`. (In a typical installation, you'll find the starter `HelloPacks` Pack at `/opt/cribl/groups/default/default/`.)

By design, you can readily share Packs **across** Fleets by copying them between Groups.

Get Started with Packs

To unpack Packs, use the above instructions (per deployment type) to navigate to the **Packs** page. Select `HelloPacks` to see its configuration.

Select **Pipelines**, and you'll see that the Pack includes `devnull`, `main`, and `passthru` Pipelines, corresponding to the default Pipelines provided at Cribl Edge's global level. This Pack also includes an Apache-specific sample Pipeline – select it to unpack that, too.

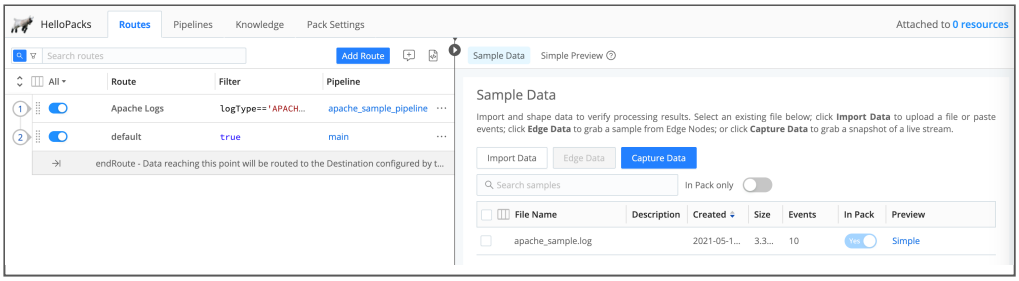


Pipelines within example Pack

Select **Routes**, and you'll see that this Pack also provides both a `default` and an Apache-specific Route.

Configure a Pack

Once loaded, each Pack displays tabs labeled with familiar links: **Routes**, **Pipelines**, **Knowledge**, and **Pack Settings** in the left pane, along with **Sample Data**, and **Simple Preview** on the right.



Configuring a Pack

The left pane tabs give you access to configuration objects specific to this Pack.

Cribl imposes a limit of 200 maximum Pipelines per Pack. This limit helps ensure optimal performance, and prevents potential errors. It helps maintain stability and avoids issues that can arise from managing very large Packs.

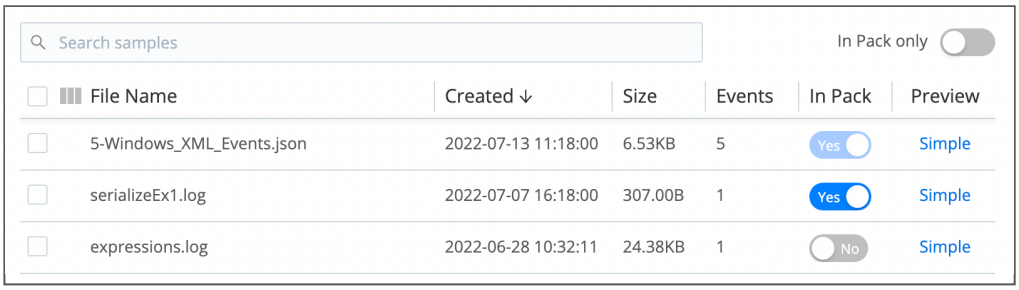
Sample Data

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**).

A grayed-out or dark-blue toggle means that this sample file is global to Cribl Edge. It is available to this Pack. Toggle this to `Yes` (dark blue) if you want the sample file to export along with the Pack.



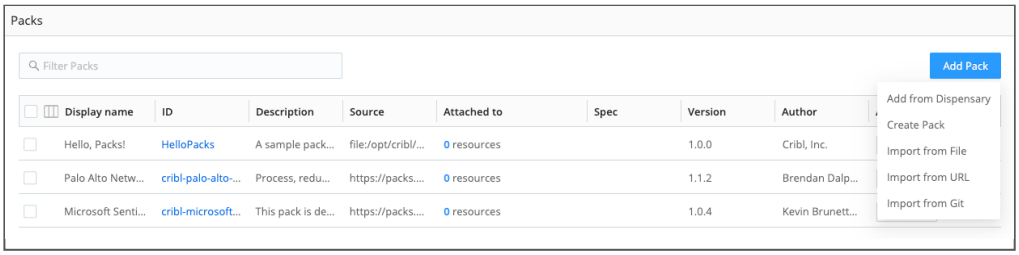
Sample file in Pack

Basically, you can manipulate all the options here as you would work with the equivalent options in the global navigation.

Import or Upgrade a Pack

To import a new Pack, or an updated version of an existing Pack, from your filesystem:

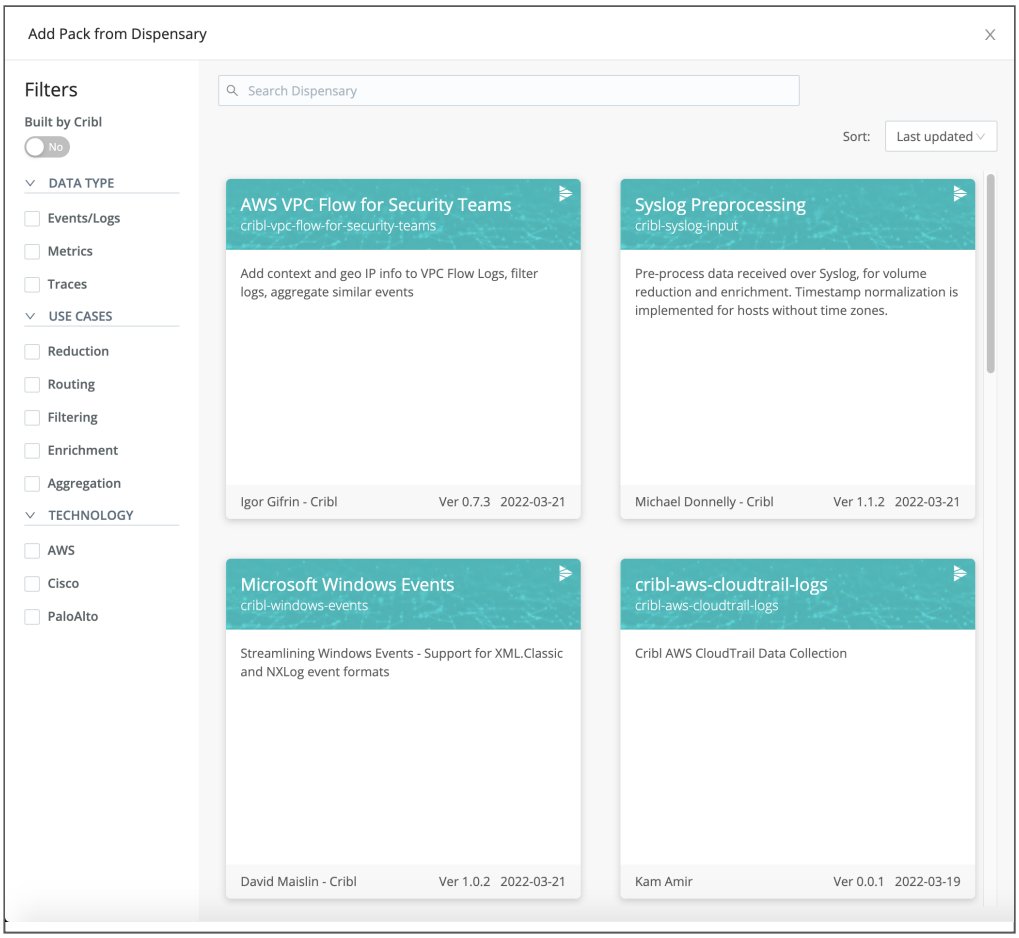
1. Navigate to the **Packs** page.
2. Select **Add Pack** at the upper right.
3. Select your desired **Add/Import** source: Dispensary, File, URL, or Git repo. When you import Packs that were exported in `merge` mode, make sure to re-enter any required secrets to restore full functionality. This is necessary because Cribl Edge deletes all encrypted fields during the export process to ensure security and prevent the accidental sharing of sensitive information.
4. Follow the above links to details on each of these options.



The Cribl Packs Dispensary

The Cribl Packs Dispensary is a Cribl-hosted resource for you to find and share Packs. Cribl, our partners, and community users develop Packs and submit them to the Dispensary for easy sharing. Cribl tests submissions before publication, to ensure each Pack's quality, security guardrails, and stability.

You can install Dispensary Packs directly through Cribl Edge's user interface, as outlined in Add from Packs Dispensary below.



Cribl Packs Dispensary (as displayed in Cribl Edge's **Add** drawer)

Interested in publishing your own Packs on the Cribl Packs Dispensary? See Publishing a Pack.

Add from the Packs Dispensary

To add a Pack from the Cribl Packs Dispensary sharing site:

1. Navigate to the **Packs** page.

2. Select **Add Pack** at the upper right and select **Add from Dispensary**.
3. The Packs Dispensary will open in a drawer, as shown in the screenshot above.
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 this page.
7. That's it! You'll see a banner confirming that the Pack is now installed.

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 below.)
5. Select **Import** to confirm the import.

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.

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

To import a Pack from a known public or private Git repo:

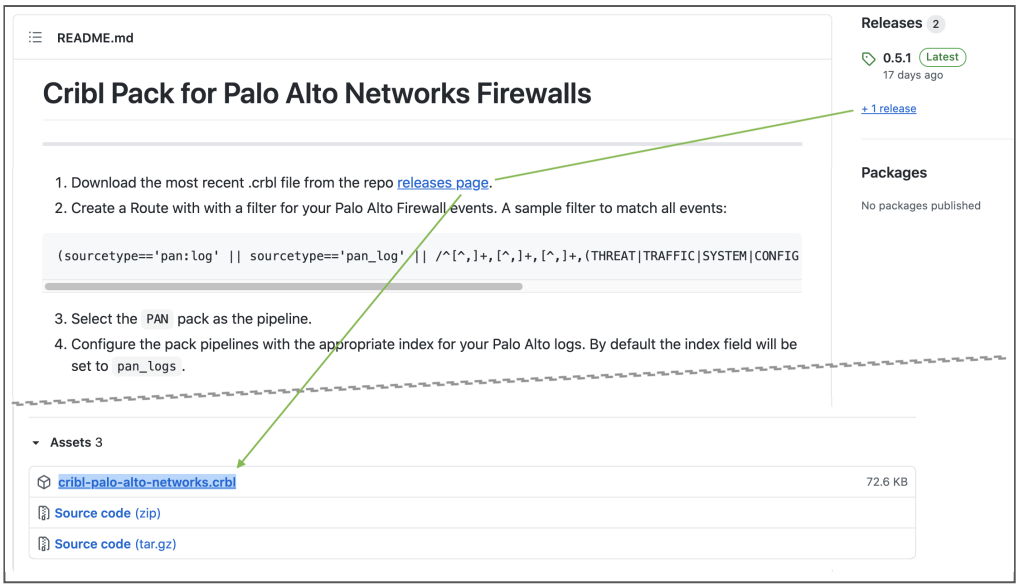
1. Navigate to the **Packs** page.
2. Select **Add Pack** at the upper right and select **Import from Git**.
3. Enter the source repo's valid URL.
   This field's input is validated for URL format, but not for completeness or accuracy, before you submit the modal. When targeting a private repo, use the format: `https://<username>:<token/password>@<repo-address>`. Public repos need only `https://<repo-address>`, as shown in the example below.
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.

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.

Dispensary GitHub Repo

A particularly useful public repo is the Cribl Pack Dispensary on GitHub. This repo was established prior to the Cribl-hosted Cribl Packs Dispensary site described above, and it is a place to collaborate on developing Packs prior to submitting them to the newer site.

You can install this repo's Packs directly through Cribl Edge's user interface, as outlined in Import from Git Repos above. However, if you prefer, you can click through to any Dispensary repo release page, download the corresponding `.crbl` file, and then upload the file into Cribl Edge.



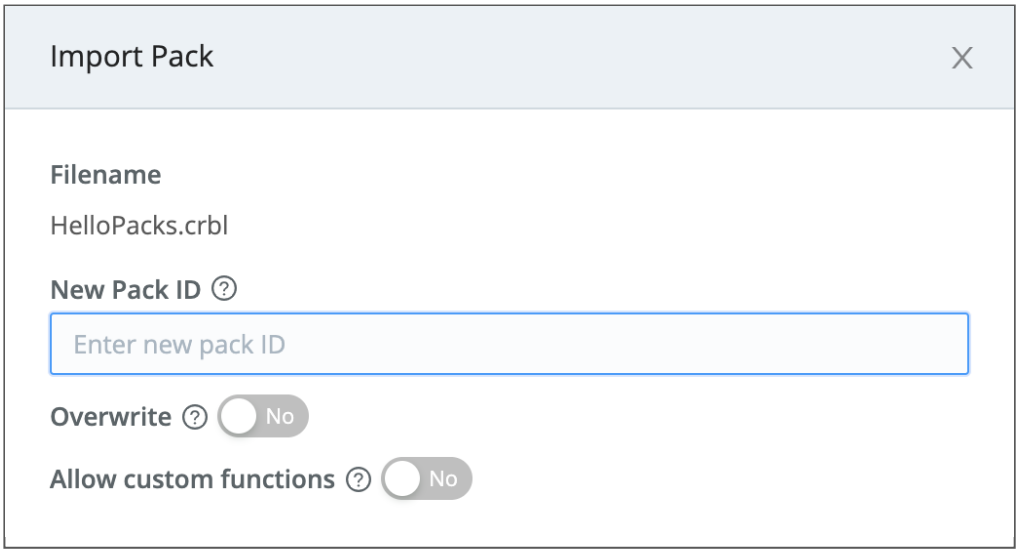
Downloading a `.crbl` file from the Cribl Pack Dispensary's Web UI

If you've posted completed Packs to our GitHub repo, we encourage you to also submit them to the Cribl Packs Dispensary site. See Publishing a Pack.

Upgrade an Existing Pack

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'll be prompted to assign a unique **New Pack ID** to import it as a separate Pack.



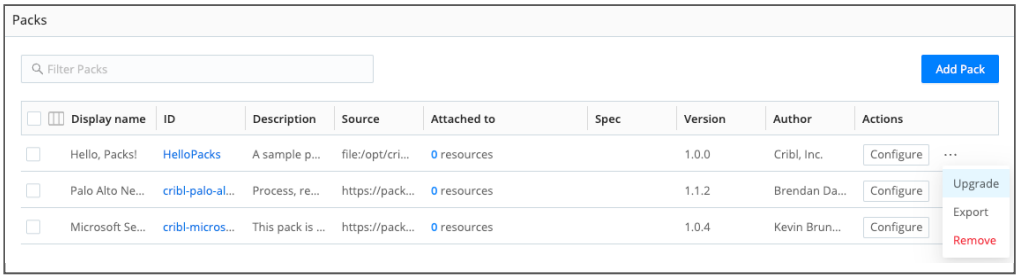
Renaming a Pack on import

You'll also have the option to **Overwrite** the installed Pack, reusing the same ID.

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 Cribl Edge instance must have a unique **Pack ID**, so you cannot share an ID between two (or more) installed Packs.

To explicitly upgrade an **existing** Pack, you can instead open the Options ••• menu on its row, and then select **Upgrade**.



Upgrading an existing Pack

If you've modified an installed Pack, Cribl Edge will block the overwrite of the Pack, to prevent deletion of your locally created resources.

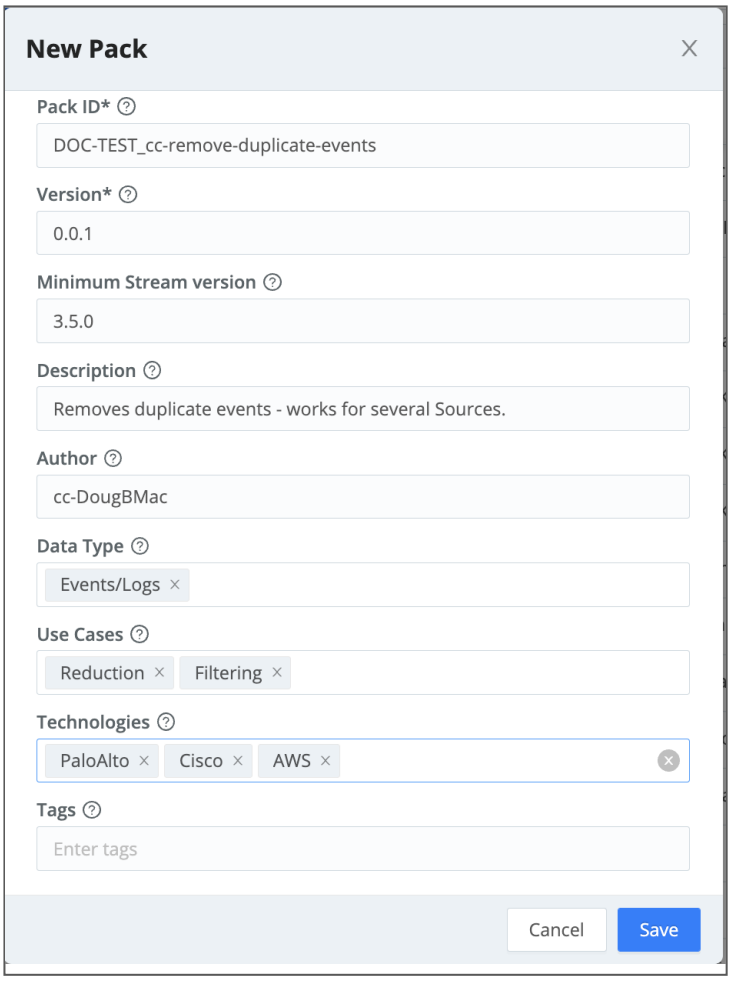
When upgrading a Pack, Cribl recommends that you:

- 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.

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 Cribl Edge instance 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 Stream 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**.



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 and save the infrastructure you want to pack up. As you save changes in the UI, they're saved to 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.

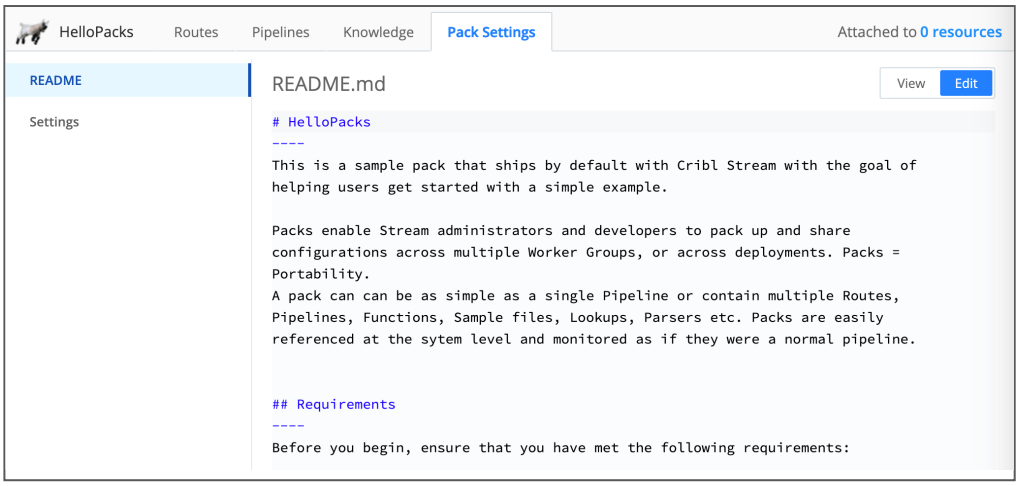
If you have a Cribl.Cloud account, you can also collaborate there by inviting team members to your Cribl.Cloud Organization. See Inviting and Managing Other Users.

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!

Modify Pack Settings

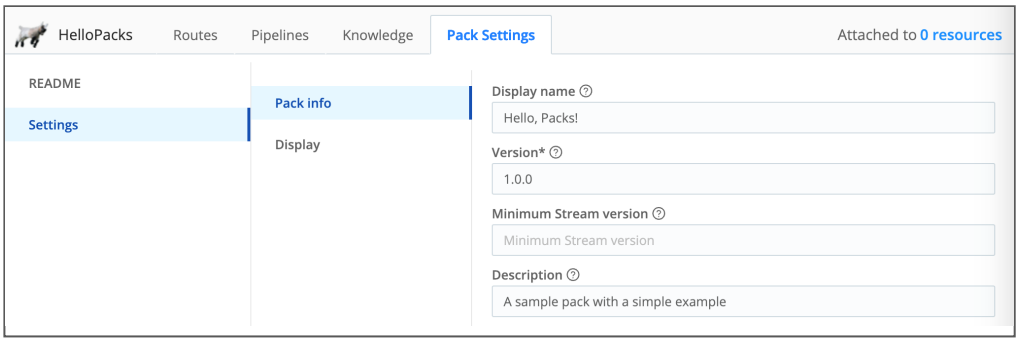
You can update Pack metadata (such as Version, Description and Author) and the Pack's display settings. If you're developing a new Pack to share, you'll want to use this interface to populate the README and display logo within the Pack:

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

3. To update other metadata, select the left **Settings** tab.



Editing a Pack's metadata

4. 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.

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 follows.

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, your first step is to select their check boxes on the **Packs** page. Then, Options ••• menu, select **Copy selected Packs to another Fleet**.

Next;

1. Select the Fleets to which you want to copy the Pack.

2. Optionally, select the **Overwrite** option to replace existing Packs that have the same **Pack ID**. (Use this option with caution, because it is destructive.)
3. Select **Copy**.

A status modal will either display a success message, or list any Packs that failed to copy.

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**: 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 **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.

Manage Packs via API

You can perform Pack operations by running Cribl Edge API calls on the command line or in a script. The API option is required if you plan to automate Pack operations, such as in a CI/CD pipeline.

In this section, we'll walk through one scenario where running API calls on the command line works well: exporting a Pack from one Fleet and installing it into another. The two Worker Groups/Fleets do **not** need to have the same Leader Node.

About the Following Examples

- The API calls here include Fleet names as path parameters.
- The `curl` commands assume that you have set the `$token` environment variable to match the value of a bearer token. Of course, this is just one option for authentication. See the Authentication topic for others; adapt the example commands to suit your chosen approach.

Export via API

Adapt and run this **Export pack** API call, using the export mode of your choice:

```
GET /api/v1/m/<worker_group_name>/packs/<pack_name>/export?mode=merge
```

Export Example

Let's export a Pack named `goat-herd` from the `default` Fleet, and use the `>` redirect to write the exported Pack to a file named `goat-herd.crbl`:

```
curl -X GET -H "Authorization: Bearer $token" 'https://stream:9000/api/v1/m/default/
```

This request returns an octet-stream attachment which is downloaded as a `.crbl` file. And there you are, you have exported your Pack.

Install via API

Installing the exported Pack in a different Fleet is a two-step process: First upload, then actually install.

Install via API – Step 1

Adapt and run this **Upload pack** API call, referencing the exported Pack file:

```
PUT /api/v1/m/<new_worker_group>/packs?filename=<pack_name>.crbl
```

Install Example – Step 1

We'll use our target Fleet name (in this example, it's `group420`). Then we need to specify the exported Pack contents as a file payload, using the `--data-binary` option to upload the binary data without modification. The `@` prefix tells `curl` that `goat-herd.crbl` is the path to the file, not the data itself.

```
curl -X PUT 'https://stream:9000/api/v1/m/group420/packs?filename=goat-herd.crbl' \
-H 'Authorization: Bearer $token' \
-H 'Accept-Encoding: gzip, deflate, br, zst' \
-H 'Connection: keep-alive' \
-H 'Content-Type: application/octet-stream' \
--data-binary '@goat-herd.crbl'
```

This request returns a JSON object of the following form:

```
{"source":"pack_name.random_id.crbl"}
```

Install via API – Step 2

Adapt and run this **Install pack** API call:

```
POST /api/v1/m/<new_worker_group>/packs
```

Meanwhile, remember that this API call will need a payload – the JSON object returned by the previous API call.

Install Example – Step 2

We'll use the `curl -d` option to specify the JSON object payload. We'll add a new element to the object, whose key is `id`, and whose value is the Pack's new name in the new Fleet.

Here, we rename the `goat-herd` Pack as `billys_pack`. (If you do not wish to rename the Pack, just omit the `id` element – but keep the `source` element.)

```
curl -X POST 'https://stream:9000/api/v1/m/group420/packs' \
-H 'Authorization: Bearer $token' \
-H 'Content-Type: application/json' \
-d '{
  "source": "goat-herd.987654321.crbl",
  "id": "billys_pack"
}'
```

Copy via API

To bulk-copy Packs between @{groups}s, adapt and run this **Clone pack** API call, referencing a source Fleet, destination Fleet(s), and Pack(s).

```
POST /api/v1/packs/__clone__

{
  "srcGroup": "copy_from_this_worker_group_id",
  "dstGroups": [
    "destination_group_1",
    "destination_group_2",
    ...
  ],
  "packs": [
    "pack_id_1",
    "pack_id_2",
    ...
  ]
}
```

Copy Example

For example, to copy the Palo Alto Networks and Cisco ASA Packs from the default to `dc1-logs` and `dc2-logs` Fleets:

```
curl -X POST -H "Authorization: Bearer $token" -H "Content-Type: application/json" '|
```

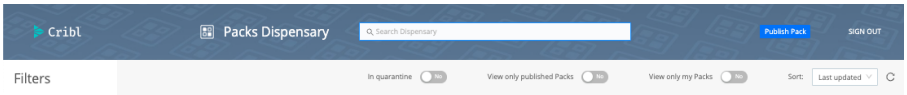
12.5.1. Packs Publication Standards

This page outlines the process for Cribl Community members to publish Cribl Edge 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 the Pack Review Summary and Pack Review Checklist below. This 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), Sources, and Destinations you're going to work with.
2. **Develop your Pack**
    - Development is generally done in your own Cribl environment, using saved samples to build out pipelines. During this stage, you might consider working collaboratively within Cribl's Community Slack `#packs` channel. When the Pack is ready for review, export the `.crbl` file using the Cribl UI.
3. **Test your Pack**
    - Cribl strongly recommends that you test the Pack yourself, in your own environment. To test that all content was exported correctly, import your Pack into a fresh Cribl environment, such as a newly created Docker instance, or to a newly created Worker Group in your Cribl environment. 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.



Packs Dispensary controls

    - 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 the Pack has been uploaded, Cribl will review your Pack. 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.

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 checklist below. Any Pack that fails to satisfy the checklist's requirements will be rejected.

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.
- Install the **Syslog Pre-processing** (`cribl-syslog-input`) Pack as a reference – it includes clear examples of each item on the checklist.

Pack Review Checklist

1. Naming

Every Pack must have both a Pack ID and a Display Name.

When you view Packs in the Dispensary, the Display Name appears above the Pack ID. For example, **Syslog Pre-processing** is a Display name while `cribl-syslog-input` is a Pack ID:

Display Name and Pack ID

Pack ID

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 the hyphen (`-`) as a delimiter.
- Should **not** include the word "pack."

For Packs normalizing to or from Elastic's Common Schema, the Pack ID (but not the Display Name) should include `-source` or `-dest`.

Display Name

The Display Name is a human-friendly version of the Pack name that:

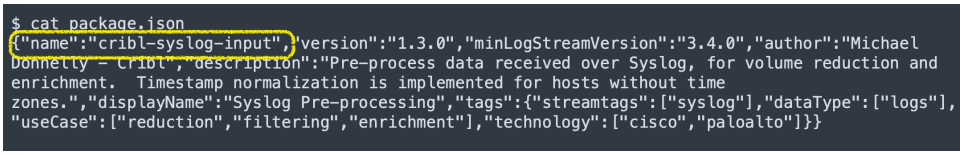
- Omits `cc-` and `cribl-`.
- Uses spaces where appropriate, and proper capitalization.
- If it includes a company or product name, matches the company's preferred spelling, spaces, and capitalization.

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

2. Pack Settings

You must review your **Pack Settings** to verify that the Pack satisfies all of the following criteria.

Display Name

A human-friendly 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.

- 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. For example, a Windows logo for Windows events, an AWS Cloudwatch logo for AWS Cloudwatch data, and so on.

You can set the logo via **Pack Settings** > **Settings** > **Display**.

3. Routes

A Pack submitted 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 datasets, each dataset should have its own Route/Pipeline pair. (For a good example, look at the PAN 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 dataset being matched.
- Pairs of Routes and Pipelines should have matching names.

4. Pipelines

A Pack submitted must include at least one Pipeline in addition to the stock Pipelines (`devnull`, `main`, `passthru`, `prometheus_metrics`, etc.).

- If the Pack deals with multiple datasets, there must be one Pipeline (and one Route) for each dataset. (For a good example, look at the PAN Pack.)
- The Pack must not use the stock Pipelines. Leave these Pipelines unmodified.
- 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. Refer to the **Syslog Pre-processing** (`cribl-syslog-input`) Pack for an example of the expected level of detail.
- 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.

5. Sample Files

The Pack must include at least one sample file for each of its Pipelines.

Sample files:

- Must **not** have an expiration date.
- Must **never** include PII, customer data, or customer hostnames.
- Must be named in such a way that it's 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 data samples, each specific to a Pipeline's Route and filter.
- Must work as expected on a clean installation of the Pack. This applies to all samples listed in the Pack – Pack developers should test this before submitting the Pack.

6. 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 deploy the Pack successfully.

The default sections for 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 listed below (sections **6a** through **6f**). The README may include additional sections as necessary.

6a. 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.

6b. 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.

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, 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.

6c. Upgrades

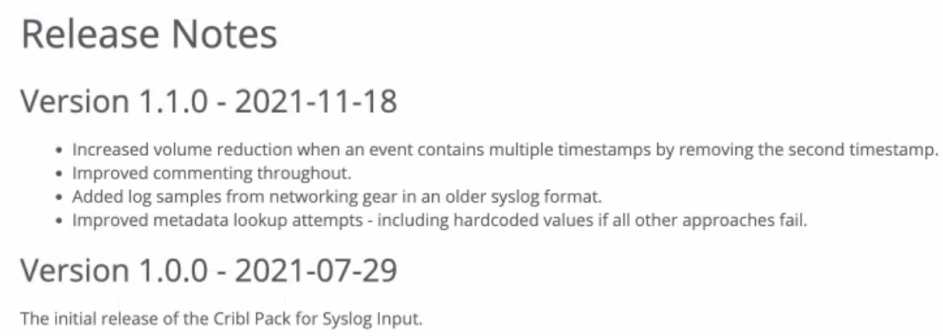
This **optional** section is strongly recommended if you expect others to customize their copy of the Pack and later upgrade the modified Pack.

- 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.

6d. 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** > **Settings**.
- Release notes versions must be listed starting with the most recent (in descending chronological order).

6e. 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
```

6f. 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:

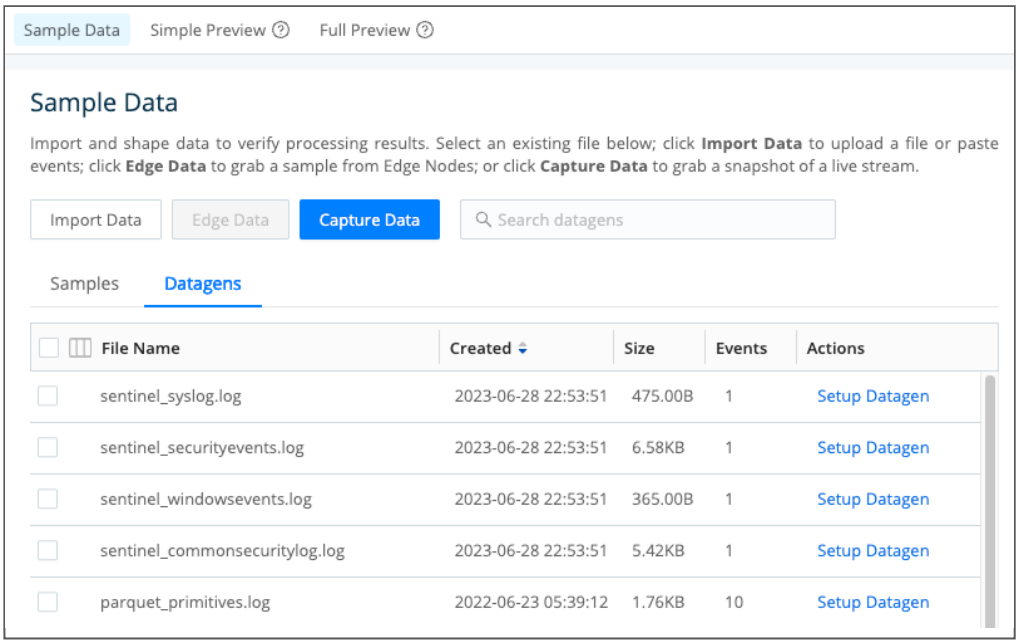
```
This Pack uses the following license: [Apache 2.0](https://github.com/criblio/ap
```

12.6. Use Datagens

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

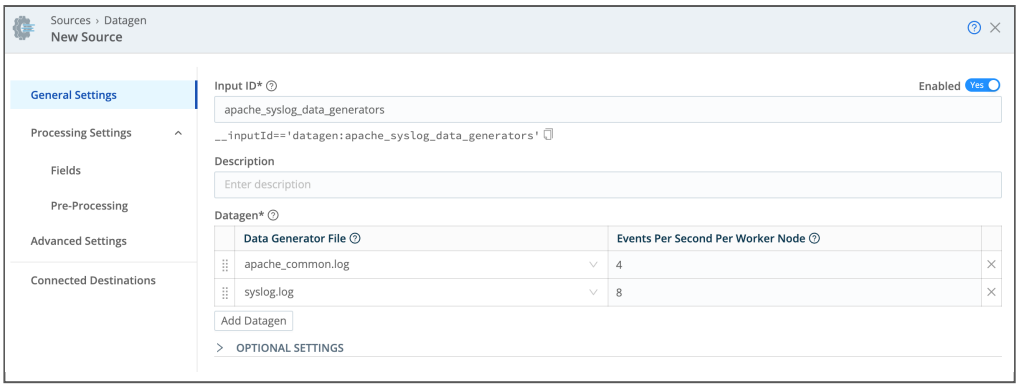
As outlined in the following tutorial: Once you've created a template, you can configure a Datagen Source to use the template to generate real-time data at a given EPS (events per second) rate.

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:

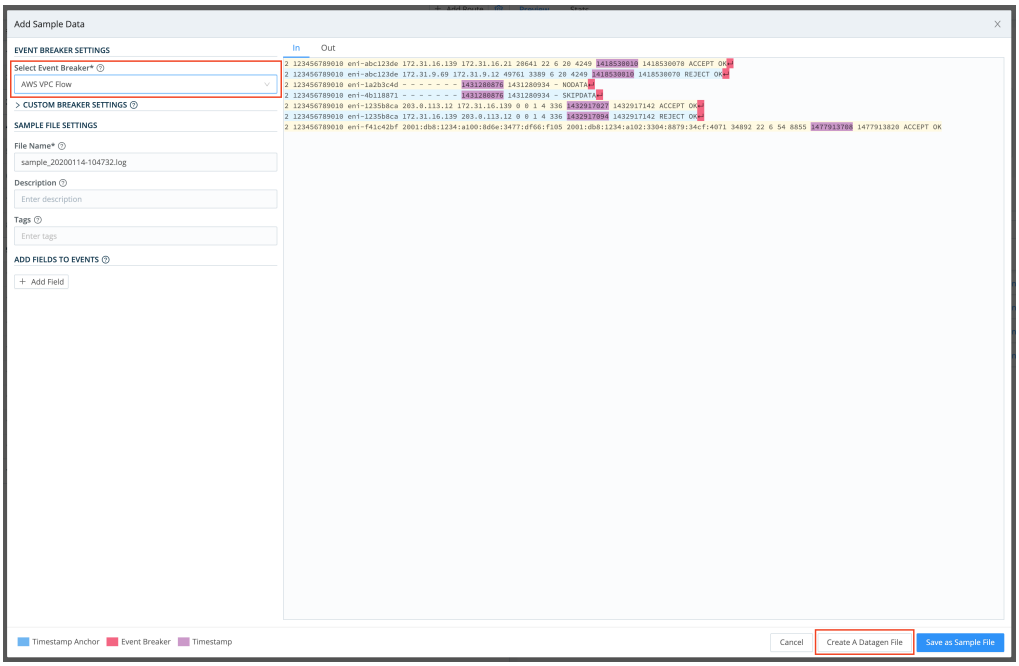
Sample VPC Flow Logs

```
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 143291
2 123456789010 eni-1235b8ca 172.31.16.139 203.0.113.12 0 0 1 4 336 1432917094 143291
2 123456789010 eni-f41c42bf 2001:db8:1234:a100:8d6e:3477:df66:f105 2001:db8:1234:a10
```

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**.

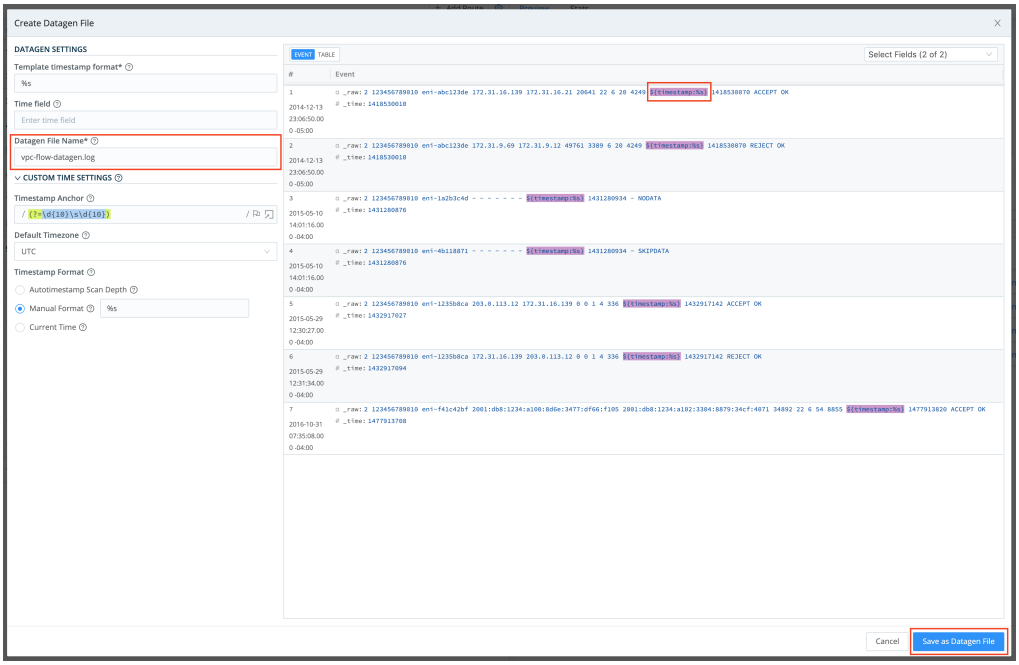


Creating a datagen template

On the resulting **Create Datagen File** screen:

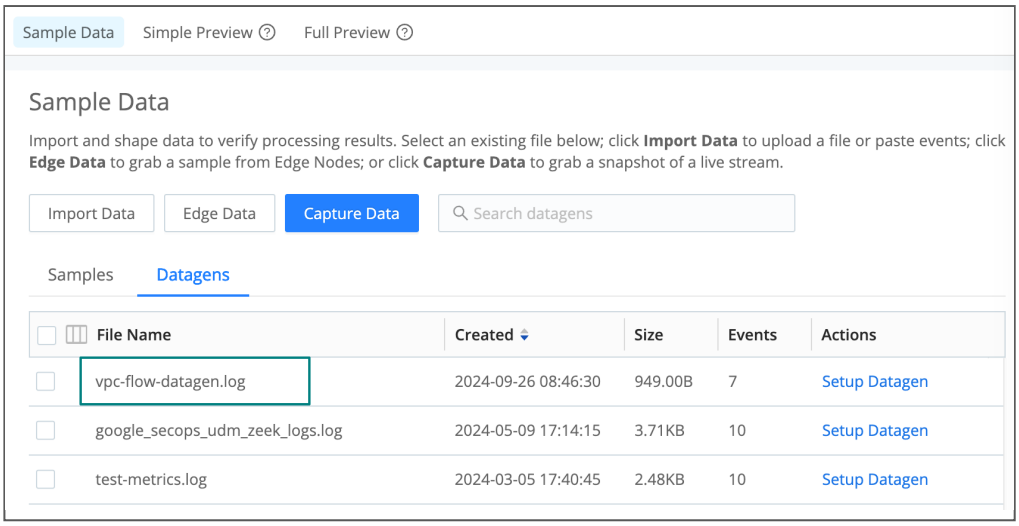
- Enter a file name, such as `vpc-flow-datagen.log`
- Ensure that the timestamp template format is correct: `${timestamp: %s}`
  `${timestamp: <format>}` is a template that the datagen engine uses to insert the current time – in each newly generated event – using the given format. In this case, `%s` is the desired `strftime` format for the timestamp (the epoch).

Once you've verified these results, click **Save as Datagen File**.



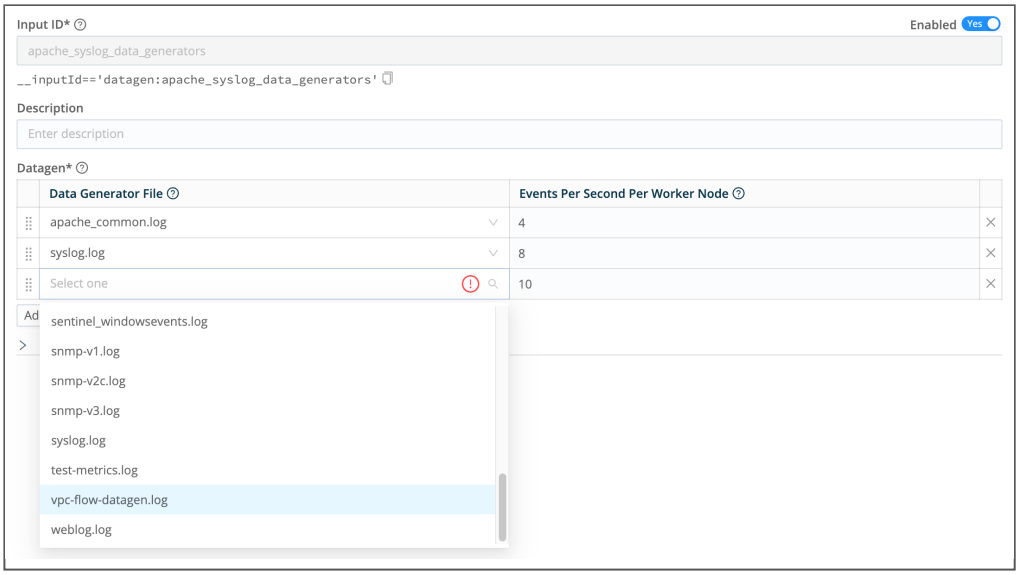
Saving a named datagen template

To confirm that the datagen file has been created, select the **Datagens** tab in the preview pane.



Verifying datagen file creation

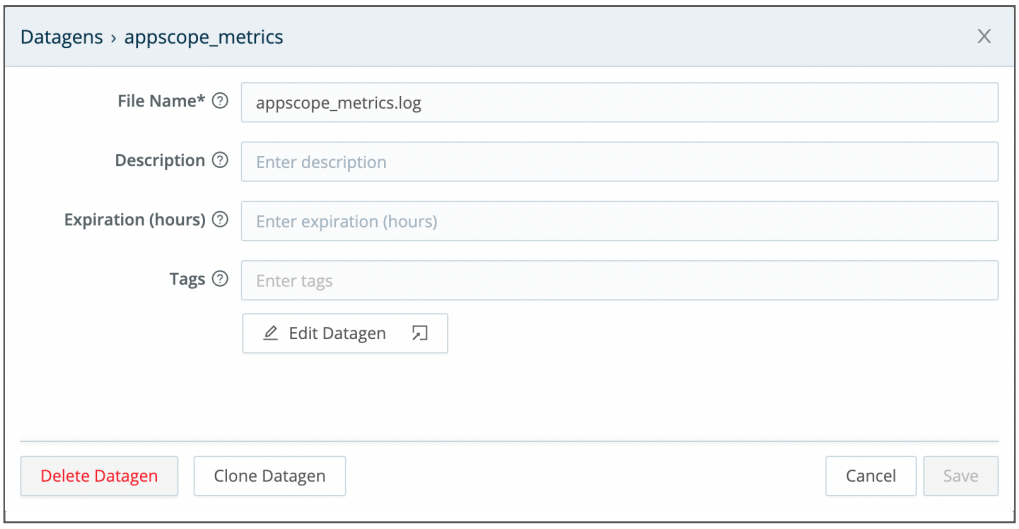
Now, to start using your newly created datagen file, go back to **Sources** > **Datagens**. Add it using the drop-down shown below.



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.

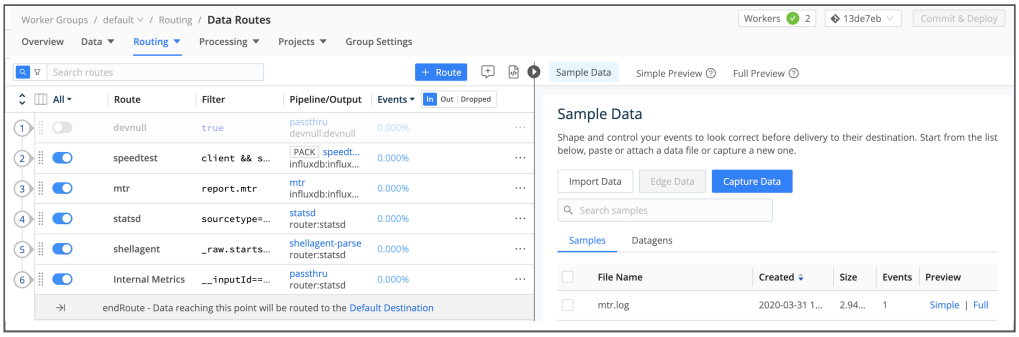
Editing a datagen

12.7. Data Preview

Cribl Edge's Sample Data Preview features enable you to visually inspect events as they flow into and out of a Pipeline. Preview helps you shape and control events before they're delivered to a Destination, and helps you troubleshoot Pipeline Functions.

Preview works by taking a set of sample events and passing them through the Pipeline, while displaying the inbound and outbound results in a separate pane. Any time a Function is modified, added, or removed, the Pipeline changes, and so does its displayed output.

The Preview pane is shown below, to the right of the Pipelines pane.

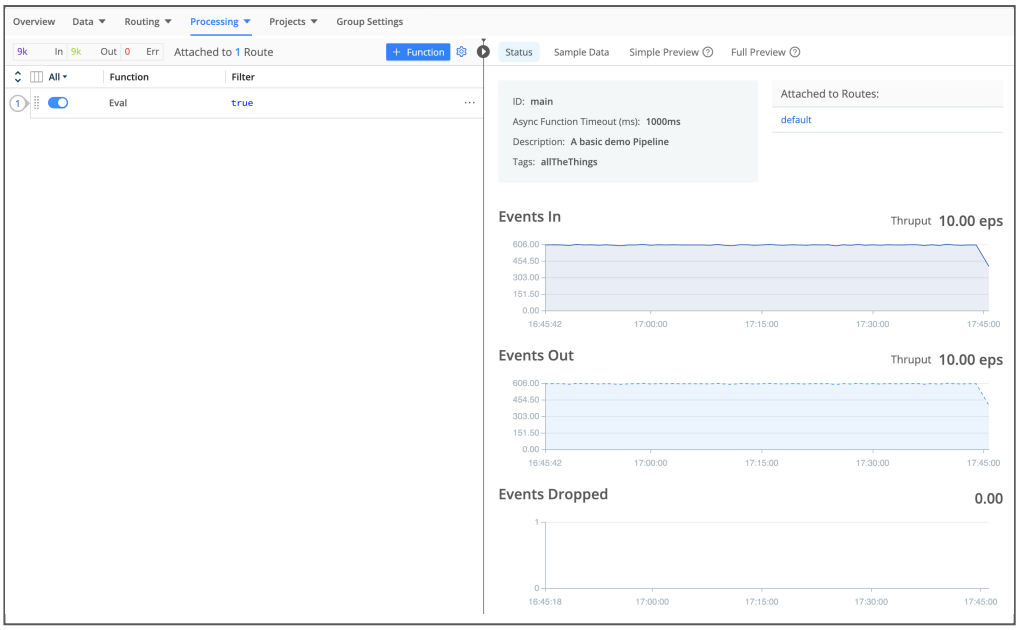


Preview options

You can press the] (right-bracket) shortcut key to toggle the visibility of the Preview pane. (This shortcut works when no field has focus.)

Check Pipeline Status

From the Pipelines page, select any Pipeline at left to display a **Status** tab in the right Preview pane. Select this tab to reveal graphs of the Pipeline's **Events In**, **Events Out**, and **Events Dropped** over time, along with summary throughput statistics (in events per second).



Pipeline status tab

Add Sample Data

When you're on the Pipelines or Routes page, you can add samples through any of the Preview pane's supported options:

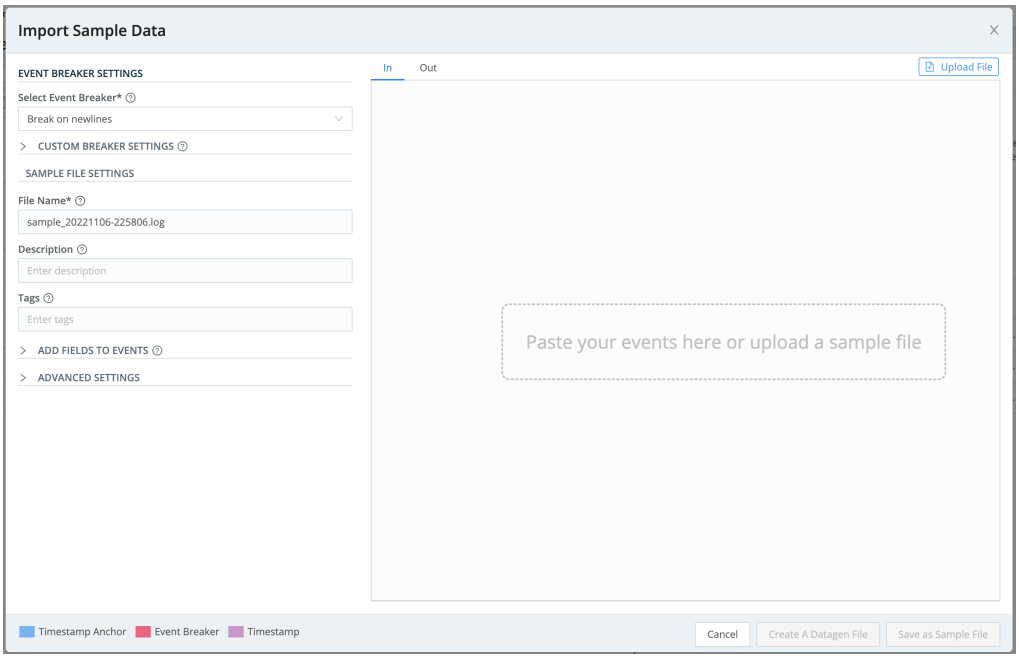
- **Import Data**: Opens a modal where you can paste data from your clipboard or use the **Upload File** button
- **Edge Data**: Lets you import data from any Edge Nodes deployed on your account
- **Capture New**

The **Import Data** options work with content that needs to be broken into events, while the **Capture New** option works with events only.

The **Edge Nodes** option requires at least one working Edge Node, and is not available when you've teleported to the Node.

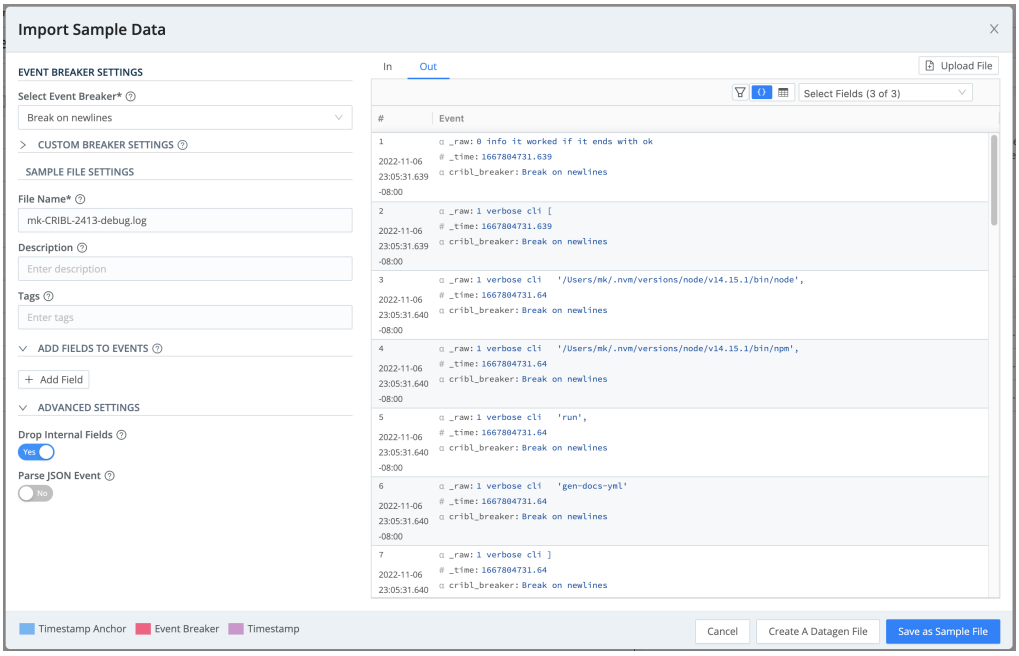
Paste Area

Once you've selected **Import Data** or imported Edge data, you'll see an **Import Sample Data** modal, where you can paste data or upload a file.



Import Sample Data modal

Once your data is onboard, you have options to modify it using Event Breakers, to add and drop fields, and to save it to a file.



Working with imported data

Edge Data

To upload data from a file on an Edge node:

1. Select **Edge Data** and navigate to the Edge node where the file is stored.
   Selecting the Edge Node opens the **Select a file** modal.

2. In **Manual** mode, use the available filters to narrow the results:
    - **Path**: Sets the location from which to discover files.
    - **Allowlist**: This filter supports wildcard syntax (as shown in the screenshot above), 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 will boldface only the top-level files within the **Path**.
3. Once you find the file you want, select its name to add its contents to the **Add Sample Data** modal, where you'll finish configuring the data sample.

Event Breaker Settings

Cribl Event Breakers are regular expressions that tell Cribl Edge how to break the file or pasted content into events. Breaking will occur 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.

Troubleshoot Event Breakers

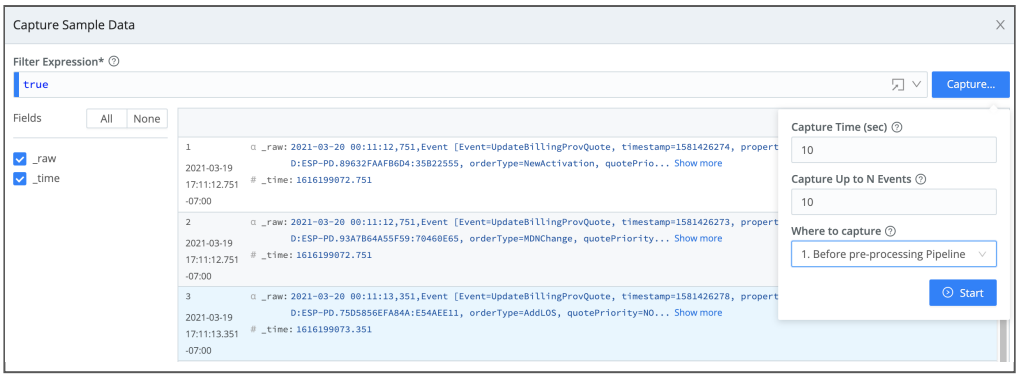
If you notice fragmented events, check whether Cribl Edge has added a `__timeoutFlush` internal field to them. This diagnostic field's presence indicates that the events were flushed because the Event Breaker buffer timed out while processing them. These timeouts can be due to large incoming events, backpressure, or other causes.

Capture Sample Data

The **Capture Data** button opens a slightly different modal – it does not require event breaking.

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.

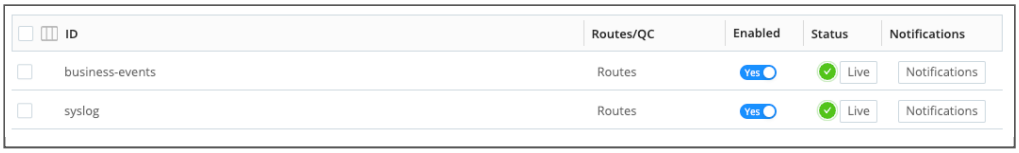
In the composite screenshot below, we've already captured some events using the **Capture** drop-down.



Capture Data > Capture Sample Data modal

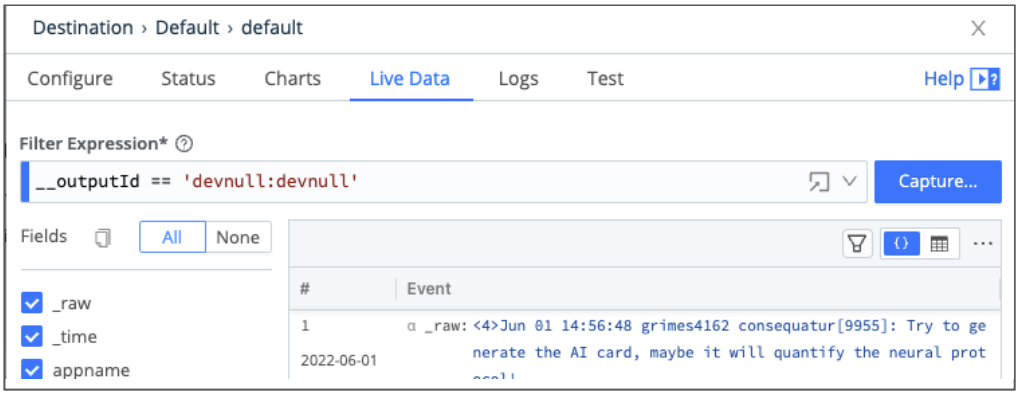
Capture from a Single Source or Destination

To capture data from a single enabled Source or Destination, it's fastest to use the Sources or Destinations UI instead of the Preview pane. You can initiate an immediate capture by selecting **Live** on the Source's or Destination's configuration row.



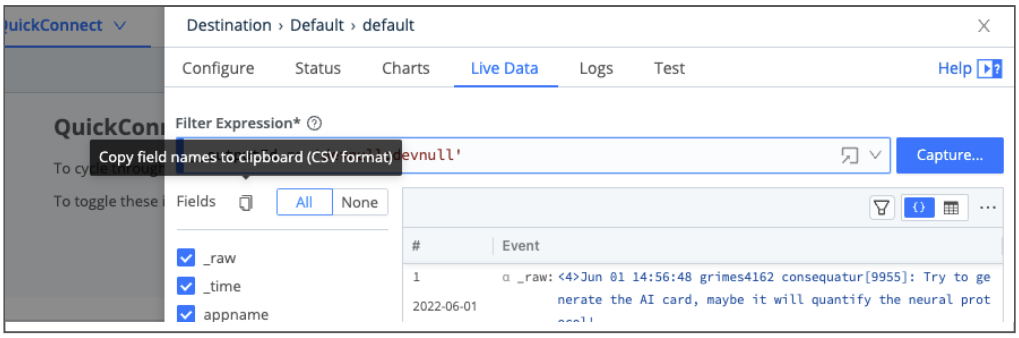
Source > Live button

You can similarly start an immediate capture from within an enabled Source's or Destination's configuration modal, by selecting the modal's **Live Data** tab.



Destination modal > Live Data tab

Beside the **Live Data** tab's **Fields** selectors 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.



Destination modal > Live Data - Copy Fields Icon

Control Sample Size

To prevent in-memory samples from getting unreasonably large, samples that you input by any means (Import Data, Edge Data, or Capture Data) are constrained by a limit set as follows:

- On a single instance at **Settings** > **General Settings** > **Limits** > **Storage** > **Max sample size**.
- On distributed Worker Groups (both Cribl-managed and customer-managed) at **Fleet Settings** > **General Settings** > **Limits** > **Storage** > **Max sample size**.

In each location, the default limit is `256 KB`. You can adjust this upward (to a maximum of `3 MB`) or downward.

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.

Fields

In the **Capture Data** and **Live Data** modals, use the **Fields** sidebar (at left) to streamline how events are displayed. You can toggle among **All** fields, **None** (to reset the display), and check boxes that enable/disable individual fields by name.

Field Type Symbols

Within the right Preview pane, each field's type is indicated by one of these leading symbols:

Symbol	Meaning
α	string

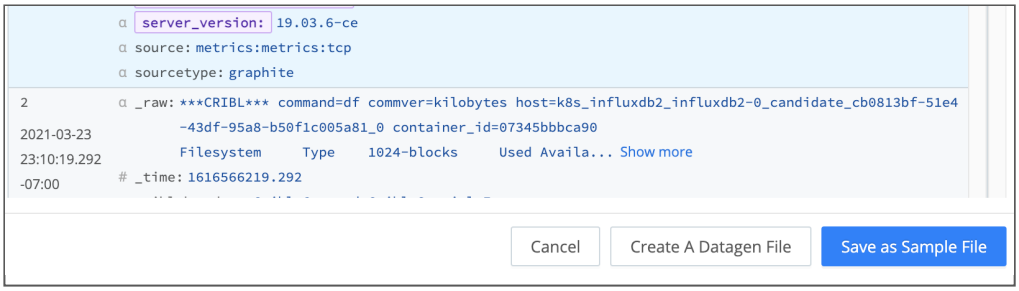
Symbol	Meaning
#	numeric
b	boolean
m	metric
{}	JSON object
[]	array

On JSON objects and arrays, you'll also see:

Symbol	Meaning
+	expandable
-	collapsible

Save Sample Data

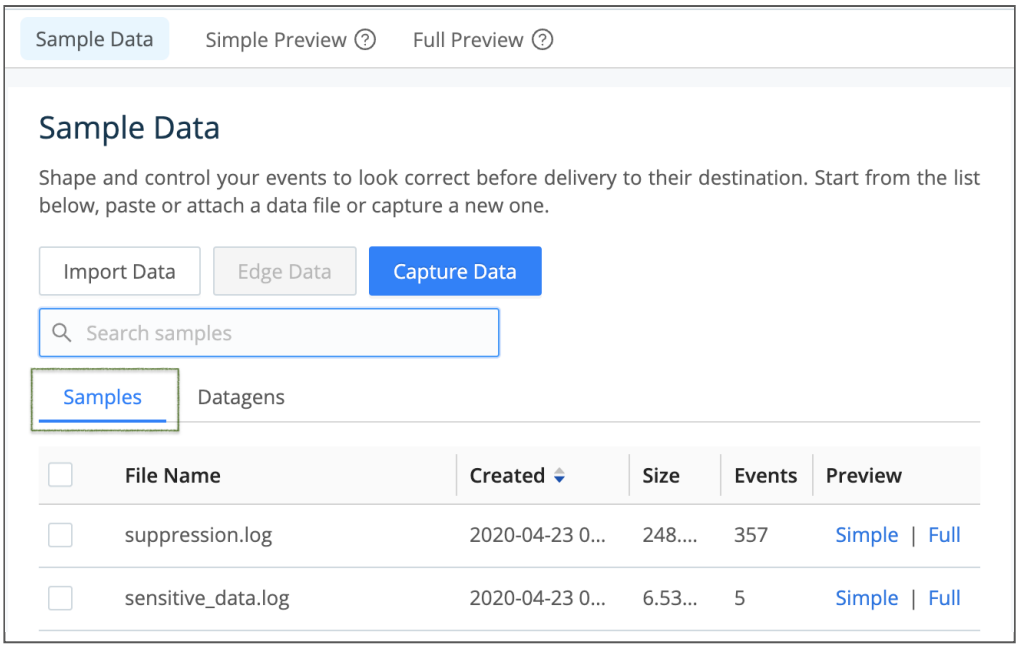
The Preview pane's **Import Data** or **Capture Data** modal, once you've successfully populated it with data, provides options to save the data as a sample and/or datagen file. Select the appropriate button, accept or modify the default/generated file name and other options, and confirm the save.



Saving sample data

Access and Manage Data Files

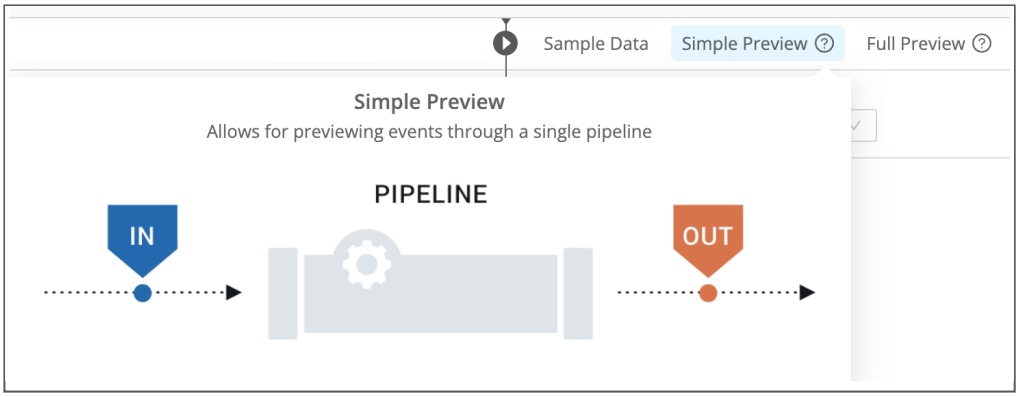
As you add more samples to your system, you can easily access them via the **Sample data file** drop-down. You can also manage and modify sample files via the **Samples** tab highlighted below.



Managing sample files

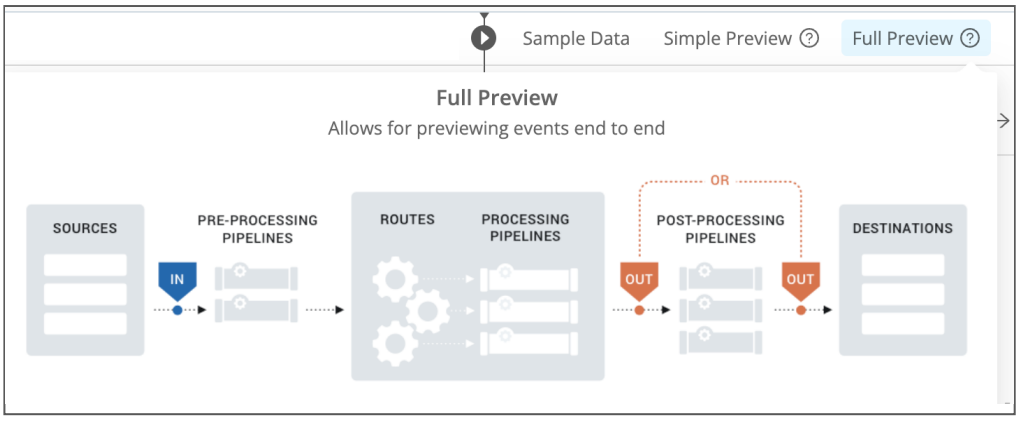
Simple Versus Full Preview

Select **Simple** or **Full** beside a file name to display its events in the Preview pane. The **Simple Preview** option enables you to view events on either the **IN** or the **OUT** (processed) side of a single Pipeline.



Simple Preview schematic

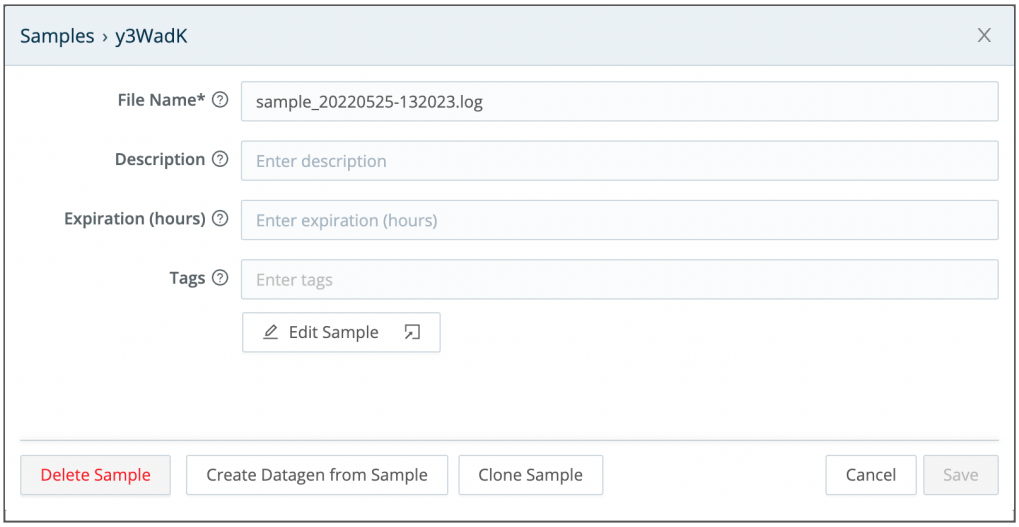
The **Full Preview** option gives you a choice of viewing events on the **OUT** side of either the processing or post-processing Pipeline. Selecting this option expands the Preview pane's upper controls to include an **Exit Point** drop-down, where you make this choice.



Full Preview schematic

Modify Sample Files

1. In the right Preview pane, select the **Samples** 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, clone, or delete the sample, save it as a datagen Source, or modify its metadata (**File name**, **Description**, **Expiration time**, and **Tags**).



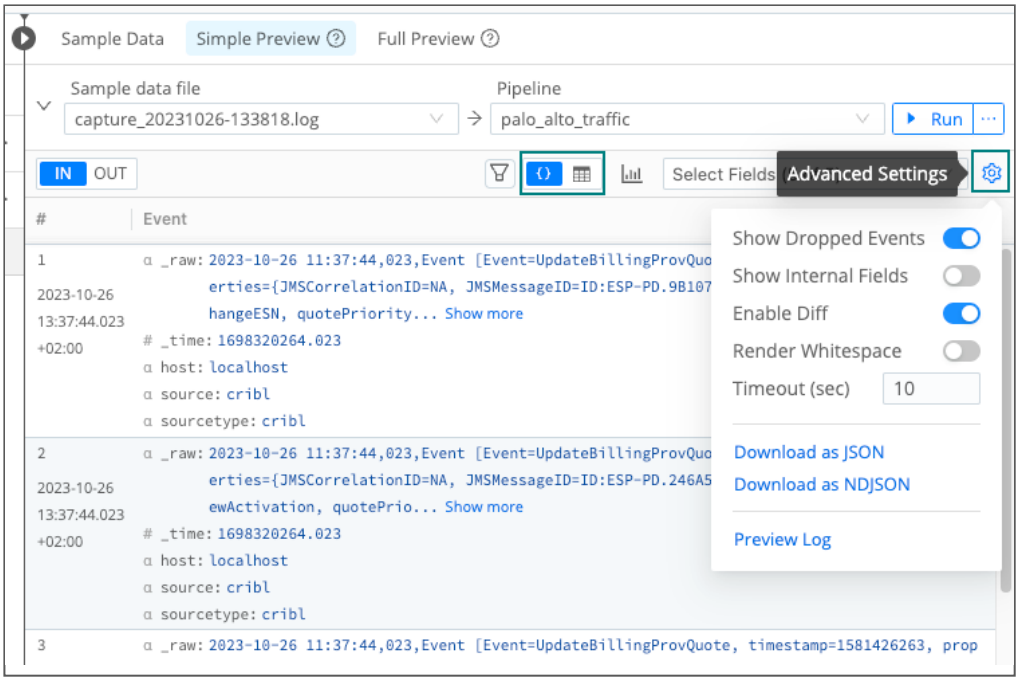
Options for modifying a sample

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

IN Tab: Display Samples on the Way IN to the Pipeline

The Preview pane offers two display options for events: Event and Table. Each format can be useful, depending on the type of data you are previewing. This screenshot shows Event view:



Event, Table, and Advanced options (composite screenshot)

On the **Advanced Settings** menu at the upper right, the first few toggles are self-explanatory, and are used primarily to filter the OUT tab's display of processed data. The following subsections cover the less-obvious controls at the menu's bottom.

Render Whitespace

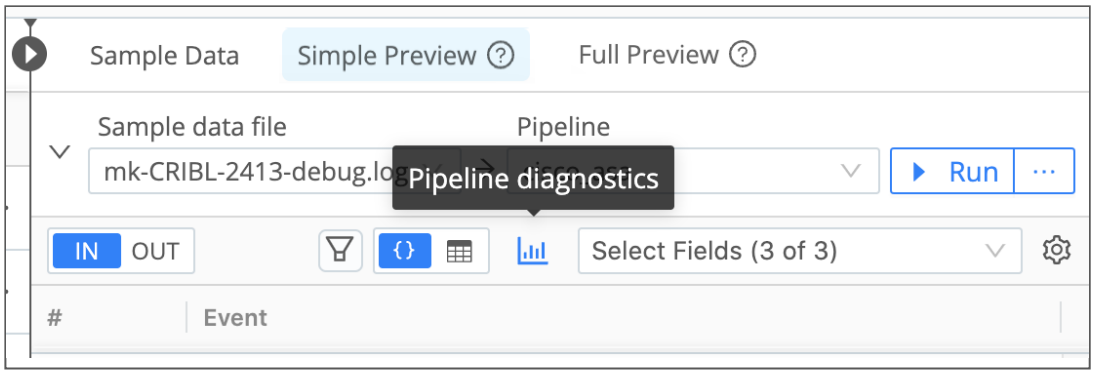
This toggles between displaying carriage returns, newlines, tabs, and spaces as white space, versus as(respectively) the symbols □, □, →, and ·.

Timeout (Sec)

If large sample files time out before they fully load, increase this field's default value of `10` seconds. A blank field is interpreted as the minimum allowed timeout value of `1` second.

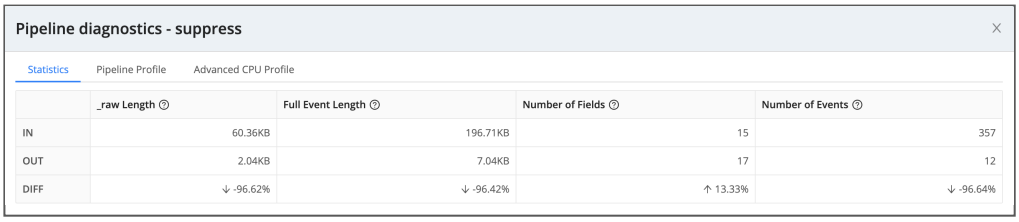
Pipeline Profiling

At the Preview pane's top, 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. (The same button is available on Pipelines within a Pack.)



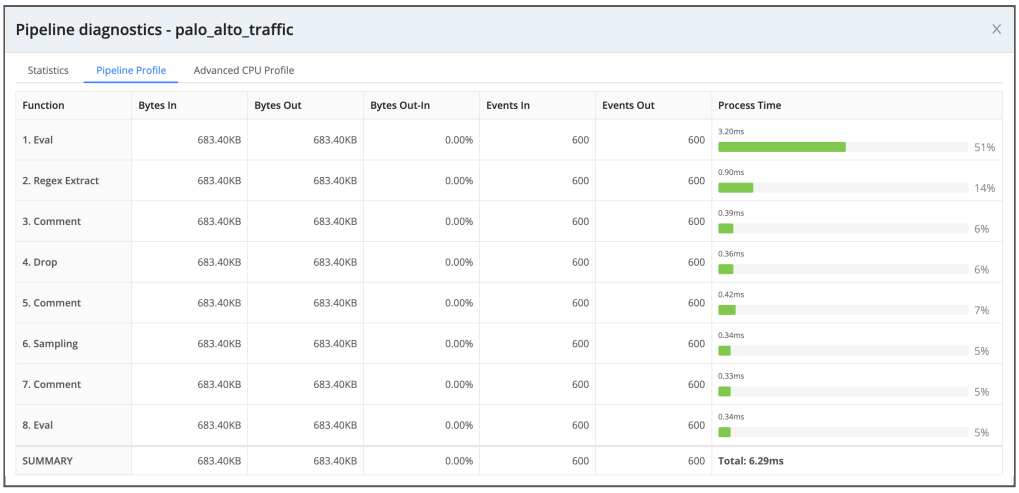
Pipeline diagnostics

The resulting modal's **Statistics** tab displays basic stats about the Pipeline's processing impact on field and event length and counts.



Pipeline diagnostics > Statistics tab

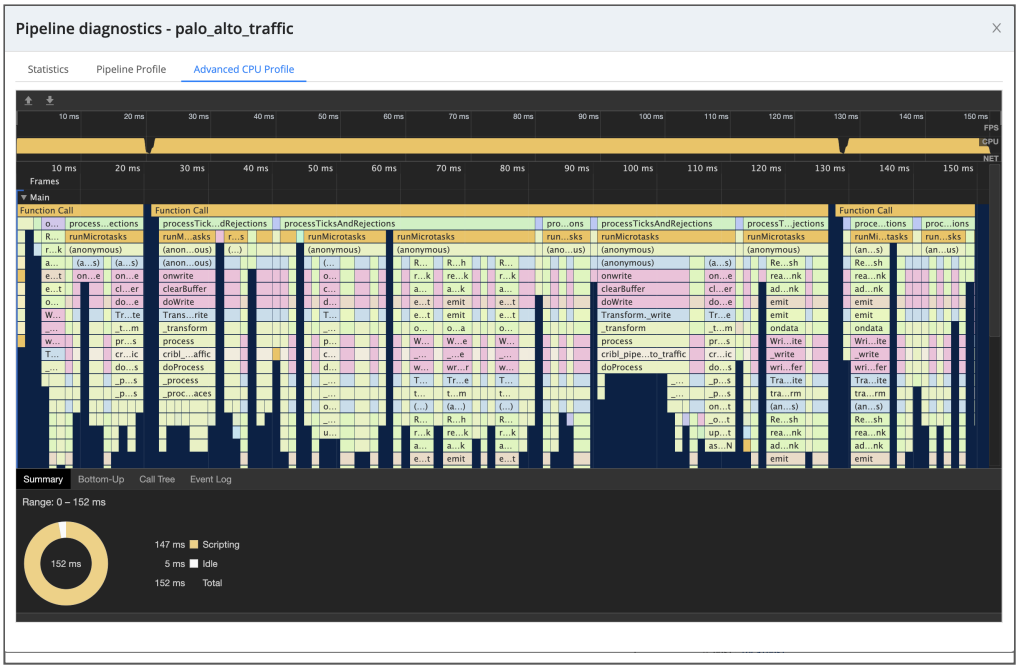
The **Pipeline Profile** tab, available from the **Simple Preview** tab, helps you debug your Pipeline by showing individual Functions' contributions to data volume, events count, and the Pipeline's processing time. Every Function has a minimum processing time, including Functions that make no changes to event data, such as the Comment Function.



Pipeline diagnostics > Pipeline Profile tab

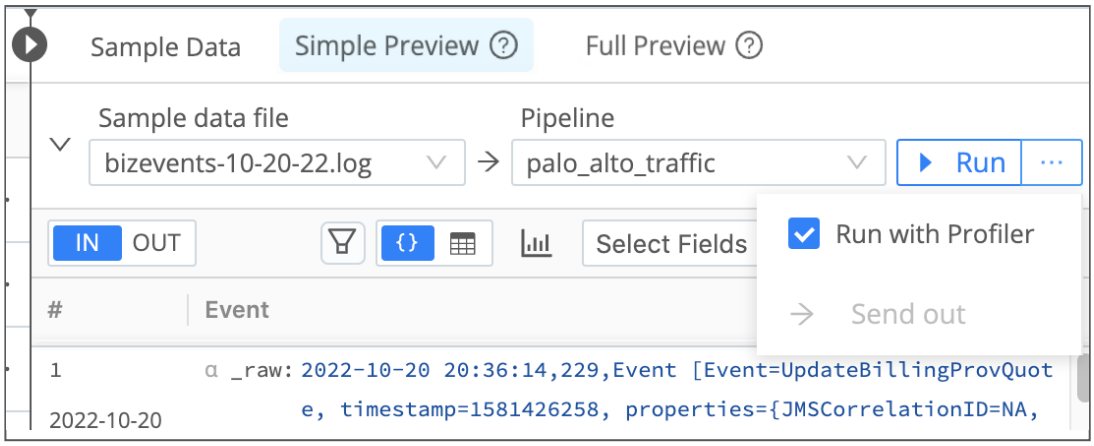
When you profile a Chain Function, this tab condenses all of the chained Pipeline's processing into a single **Chain** row. To see individual statistics on the aggregated Functions, profile the chained Pipeline separately.

The **Advanced CPU Profile** tab, available from the **Simple Preview** tab, displays richer details on individual Function stacks.



Pipeline diagnostics > Advanced CPU Profile tab

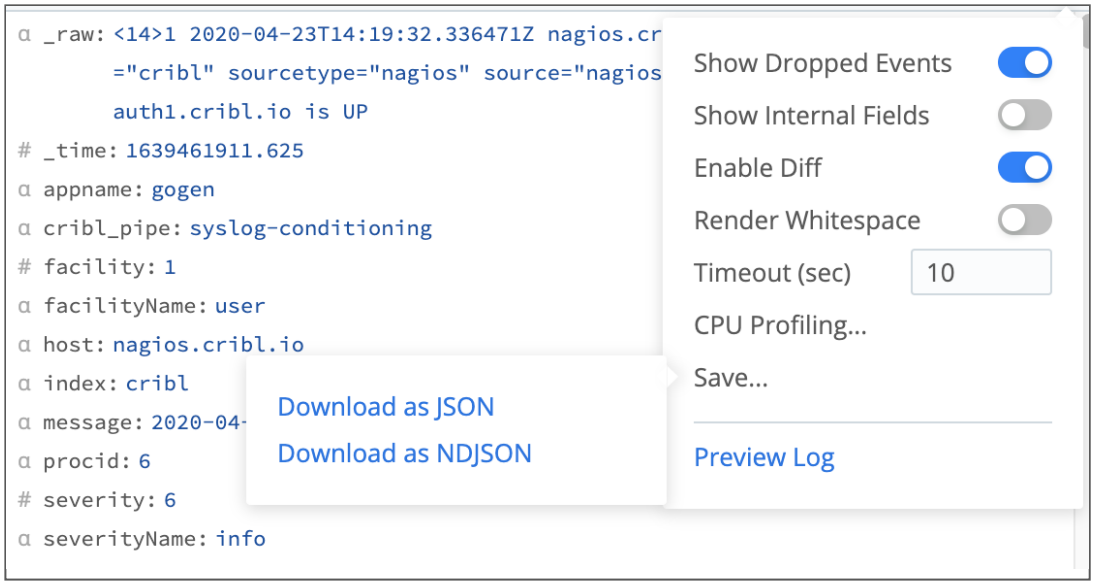
You can disable the profiling features – and regain their own (minimal) resources – by selecting the drop-down list beside the **Run** button and clearing the **Run with Profiler** check box.



Disabling Pipeline profiling

Save

The **Save** submenu enables you to save your captured data to a file, using either the **Download as JSON** or the **Download as NDJSON** (Newline-Delimited JSON) option.



Saving sample data as JSON

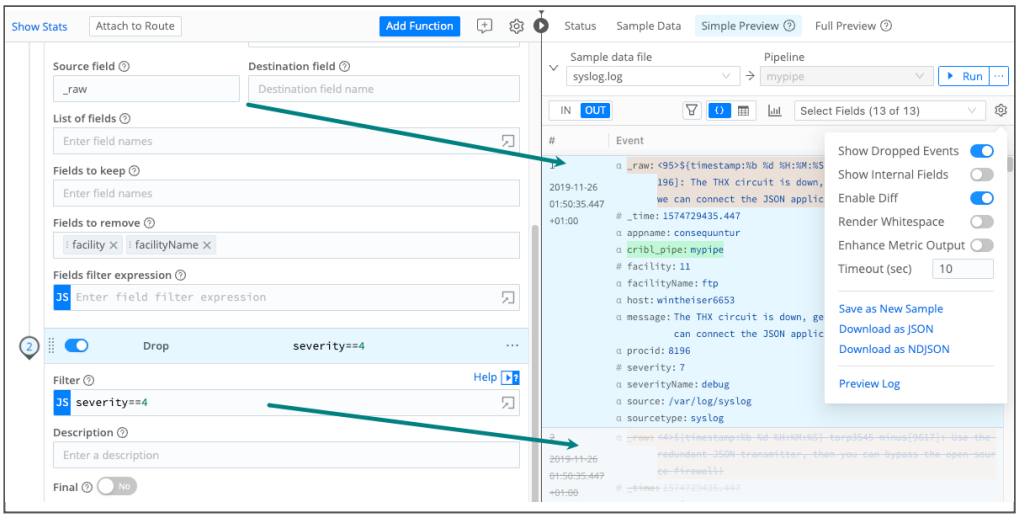
Preview Log

The final option on the **Advanced Settings** menu opens a modal where you can preview Cribl Edge's internal logs summarizing how this data sample was processed and captured.

OUT Tab: Display Samples on the Way OUT of the Pipeline

As data traverses through a Pipeline's Functions, events can be modified, and some might be dropped altogether. The **OUT** tab indicates changes using this color coding:

- **Dropped events**: When events are dropped, the **OUT** tab displays them as grayed-out text, with strikethrough. You can control their display using the **Advanced Settings** menu's **Show Dropped Events** toggle.
- **Added fields**: When Cribl Edge's processing adds new fields, these fields are highlighted green. You can control these fields' display using the **Select Fields** drop-down.
- **Redacted fields**: These fields are highlighted amber.
- **Deleted fields**: These fields are highlighted red.



Dropped and added fields in a Pipeline's output

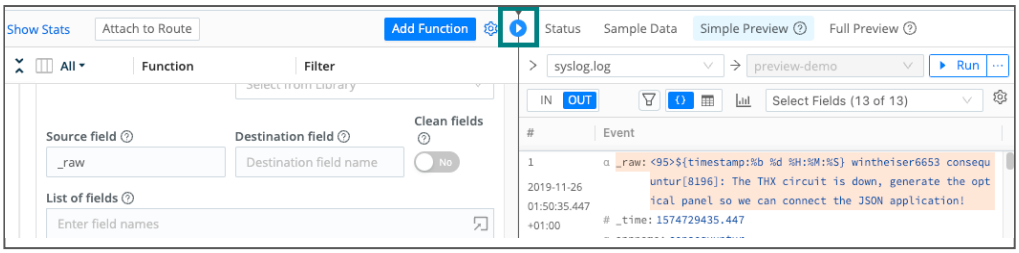
The **OUT** tab displays the same Event versus Table buttons as the IN tab. It also displays the same **Advanced Settings** menu options – and here, you can use the menu's **Show Dropped Events**, **Show Internal Fields**, and **Enable Diff** toggles to clarify how the data has been transformed by the Pipeline.

Enable **Show Internal Fields** to discover fields that Cribl Edge adds to events, as well as Source-specific fields that Cribl Edge forwards from upstream senders.

Use **Enhance Metric Output** to control how the OUT tab presents the **Metric name expression** resulting from the Publish Metrics Function. When the setting is disabled (default), the **Metric name expression** is represented as literal expression string. Enable the setting to see the expression evaluated in the OUT tab.

Manage the Preview Pane

On the pane divider between Route or Pipeline and the preview, select Collapse to hide the Preview pane. This allows the Route or Pipeline configuration to expand to your browser's full width. (The Preview pane collapses automatically on narrow viewports.)



Collapse / Expand toggle

Select Expand at your browser's right edge to restore the split view. The pane divider will snap back to wherever you last dragged it.

12.8. Data Onboarding

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.

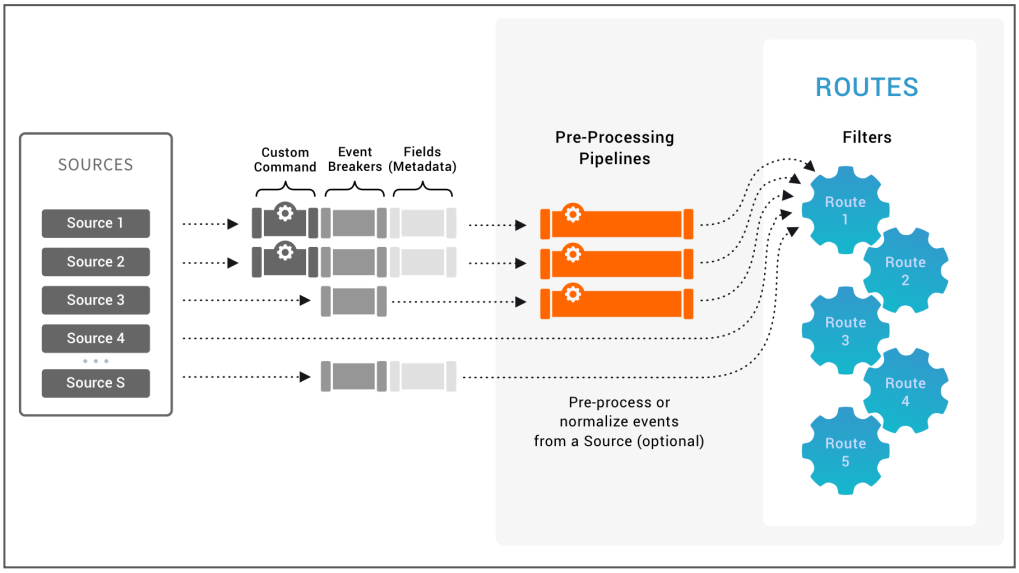
You can also obtain sample data in a live data capture from a Source. One way to do this **before** going to production is to configure your Source with a **devnull** Pipeline (which just drops all events) as a pre-processing Pipeline. Then, let data flow in for just long enough to capture a sufficient 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.

13. Functions

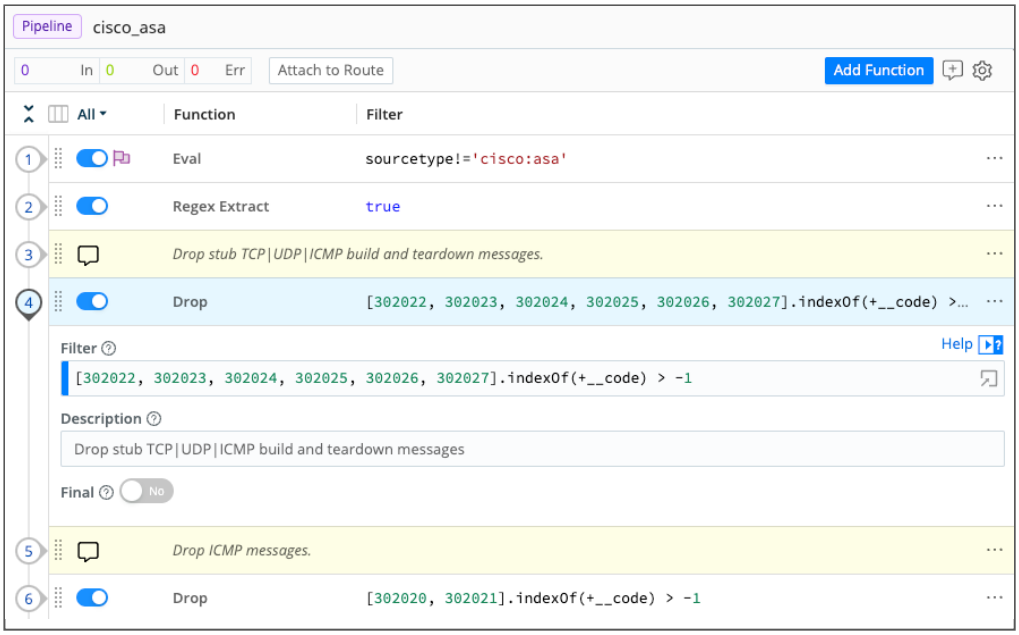
When events enter a Pipeline, they're processed by a series of Functions. At its core, a Function is code that executes on an event, and it encapsulates the smallest amount of processing that can happen to that event.

The term "processing" means a variety of possible options: string replacement, obfuscation, encryption, event-to-metrics conversions, etc. 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, Functions can be configured with filters to further scope their invocation to matching events only.

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.

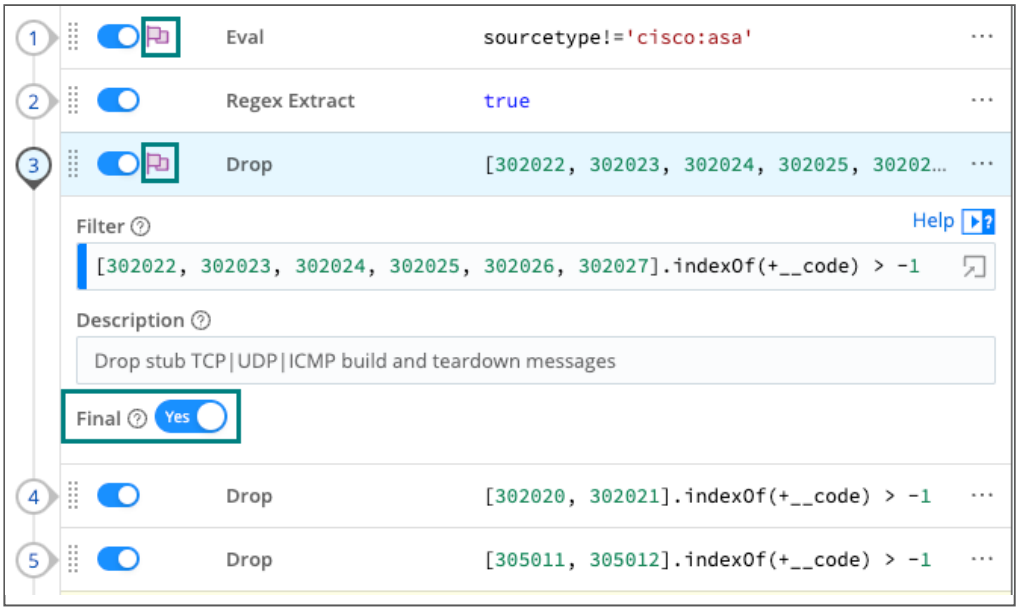
The `Final` Toggle

The `Final` toggle in Function settings controls what happens to the results of a Function.

When `Final` is toggled to `No` (default), events will be processed by this Function, **and** then passed on to the next Function below.

When `Final` is toggled to `Yes`, 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, etc.

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']`. Functions use `__e` extensively. You also **must** use this notation for fields that contain a special character, like `-`, `.`, or `@`.

Supported JavaScript Standard

Cribl Edge supports the ECMAScript® 2015 Language Specification.

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.

What Functions to Use When

- Add, remove, update fields: Eval, Lookup, Regex Extract
- Find & Replace, including basic `sed`-like, obfuscate, redact, hash, etc.: 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 (e.g, high-volume, low-value data): Sampling, Dynamic Sampling
- Suppress events (e.g, duplicates, etc.): Suppress
- Serialize events to CEF format (send to various SIEMs): CEF Serializer
- Serialize / change format (e.g., 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 (e.g., nested JSON): Flatten
- Aggregate events in real-time (i.e., 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)
- 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

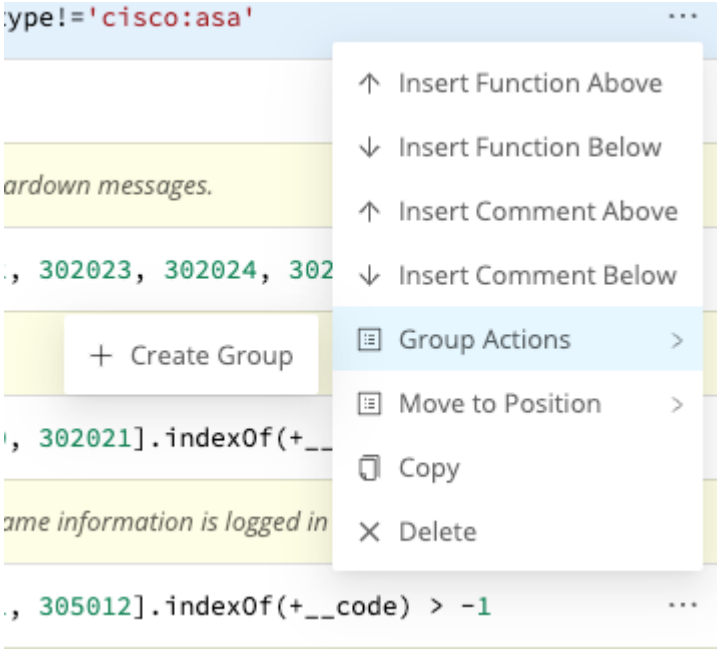
For more usage examples, download Cribl's Stream Cheat Sheet (Quick Reference Card).

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: While Functions are in a group, those Functions maintain their global position order in the Pipeline.

Function groups work much like Route groups.

To build a group from any Function, click 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 and drop it into, or out of, a group. A saved group that's empty displays a dashed target into which you can drag and drop Functions.

Drag-and-drop target

13.1. 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 the Event Breaker Function and the C.Time.timestampFinder() native method.

Usage

Filter: Filter expression (JS) 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: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

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, click **Add Timestamp** to define each format. Each row will provide these fields:

- **Regex**: Regex, with first capturing group matching the timestamp.
- **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 - milliseconds since UNIX epoch.
%s - seconds since UNIX epoch.
```

```
%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. E.g., 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

Filter: `name.startsWith('kumquats') && value=='specific string here'`

This will allow the Auto Timestamp Function to act only on events matching the specified parameters.

Sample event:

```
Sep 20 12:03:55 PA-VM 1,2019/09/20 13:03:58,CRIBL,TRAFFIC,end,2049,2019/09/20
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,2018/09/20
13:03:58,574326,1,53722,8088,53722,8088,0x400064,tcp,allow,296,296,0,4,2018/09/20
```

`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`

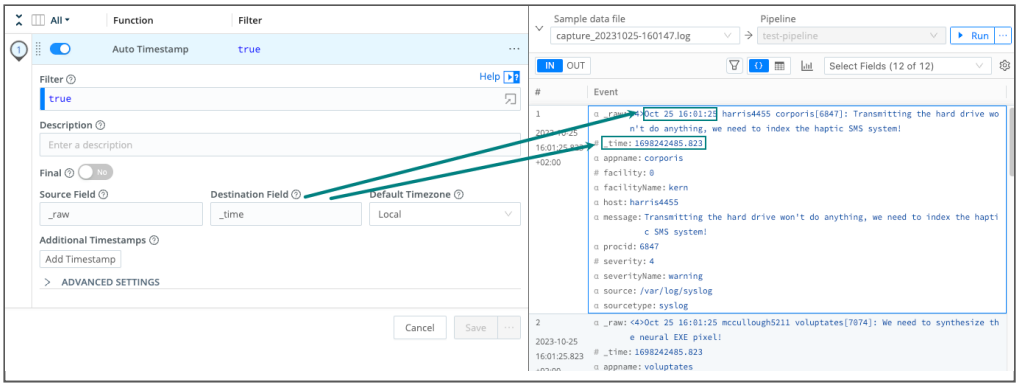
To add this sample (after creating an Auto Timestamp Function with the above **Filter** expression): Go to **Preview** > **Add a Sample** > **Paste a Sample**, and add the data snippet above. Do not make any changes to timestamping or line breaking, and select **Save as Sample File**.

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 1698242485.823`
- **GMT**: Oct 25 14:01:25 GMT
- **Your Local Time**: Oct 25 16:01:25 GMT +02:00

Because no explicit timezone has been set (under **Default Timezone**), `_time` will inherit the **Local** timezone, which in this example is `GMT +02:00`.



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.

13.2. Aggregations

The Aggregations Function performs aggregate statistics on 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 (JS) 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: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

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 there are conflicts, the last aggregate wins. For example, given two aggregates – `sum(bytes).where(action=='REJECT')` and `sum(bytes)` – the latter one (`bytes_sum`) is the winner.

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: If enabled, aggregations will be retained for cumulative aggregations when flushing out an aggregation table event. When set to `No` (the default), aggregations will be reset 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: Determines whether to pass through the original events along with the aggregation events. Defaults to `No`.

Metrics mode: Determines whether to output aggregates as metrics. Defaults to `No`, causing aggregates to be output as events.

Sufficient stats mode: Determines whether to output *only* statistics sufficient for the supplied aggregations. Defaults to `No`, meaning output richer statistics.

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 memory usage limit to impose upon aggregations. Defaults to unlimited (i.e., the amount of memory available in the system). Accepts numerals with multiple-byte units, like KB, MB, GB, etc. (such: as `4GB`.)

Flush on stream close: If set to `Yes` (the default), aggregations will flush when an input stream is closed. If set to `No`, 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
    }
  }
}
```

- `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
    }
  }
}
```

- `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.

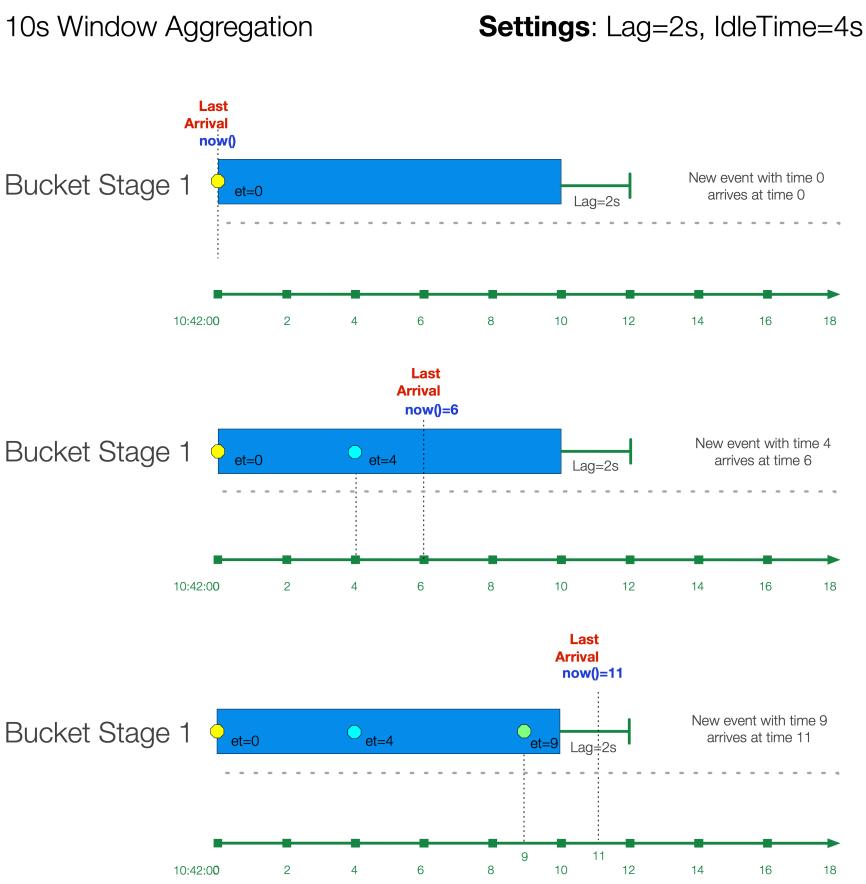
This is especially necessary for high-cardinality data. (Both settings default to unlimited, but we recommend setting defined limits based on testing.)

How Do Time Window Settings Work?

Lag Tolerance

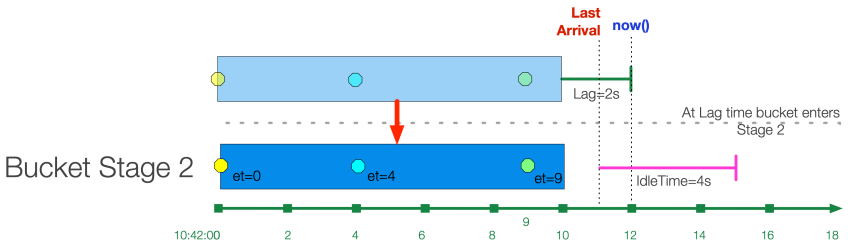
As events are aggregated into windows, there is a good chance that most will arrive later than their event time. For instance, given a `10s` window (`10:42:00 - 10:42:10`), an event with timestamp `10:42:03` might come in 2 seconds later at `10:42:05`.

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.

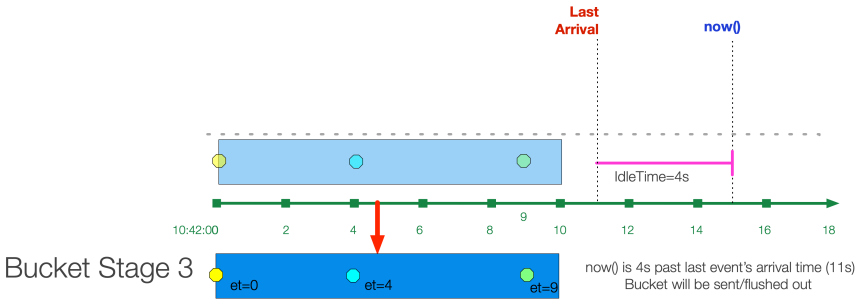
After the lag time expires, the bucket moves to Stage 2.



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`.

Time Window: `10s` Aggregations:
`sum(bytes).where(action=='REJECT').as(TotalBytes)`
`distinct_count(srcaddr).where(action=='REJECT')`

For further examples, see Engineering Deep Dive: Streaming Aggregations Part 2 – Memory Optimization.

13.3. 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 (JS) 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: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

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 JS 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

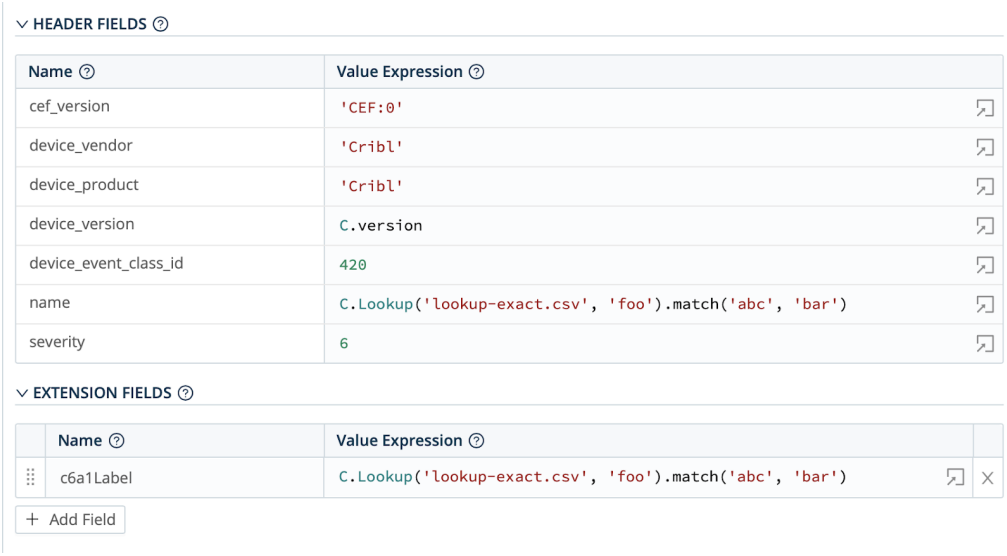
CEF Extension field definitions. Field names and values will be written in `key=value` format. Select each field's Name from the drop-down list. Values can be computed with JS expressions, or can be constants.

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
```

13.4. Chain

The Chain Function does one thing: It 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 later, Cribl Edge detects cycles when you configure your Pipeline. This early detection (compared to earlier 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 (JS) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

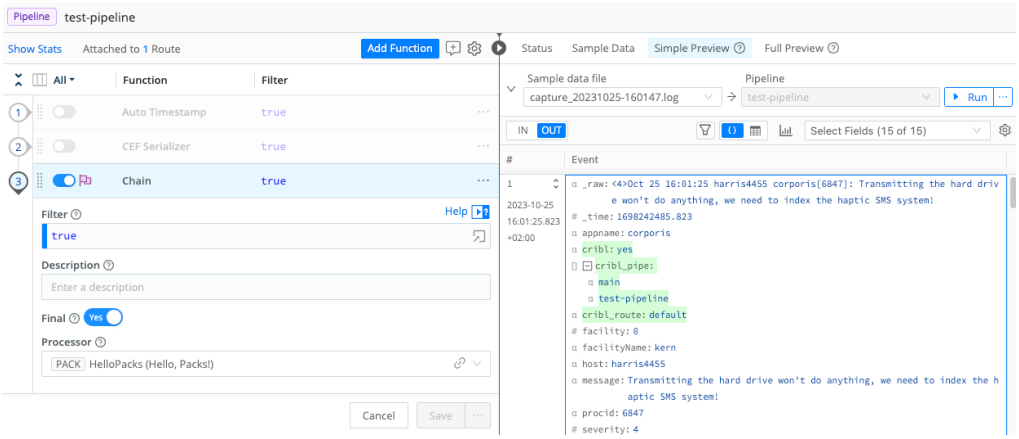
Description: Optionally, add a simple description of this Function's purpose. Defaults to empty.

Final: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`. (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

13.5. 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 (JS) 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: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

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.

13.6. Code

If you need to operate on data in a way that can't be accomplished with Cribl Edge's out-of-the-box Functions, the Code Function enables you to encapsulate your own JavaScript code. This Function is available in Cribl Edge 3.1+, and imposes some restrictions for security reasons.

Restrictions

Generally speaking, anything forbidden in JavaScript strict mode is forbidden in the context of the Code Function. Specifically, the following are **not allowed**:

- `console`, `eval`, `uneval`, `Function (constructor)`, `Promises`, `setTimeout`, `setInterval`, `global`, `globalThis`, `window`, and `set`.

Code Functions **can** include `for` loops, `while` loops, and JavaScript methods such as `map`, `reduce`, `forEach`, `some`, and `every`. For further details, see Supported JavaScript Options.

Cribl Edge's predefined Functions, such as Eval, cover the vast majority of scenarios that users typically need to implement. You should use Code Functions only as a last resort, when you need to construct a complex block of code.

Also, only skilled JavaScript developers should define Code Functions. This is to avoid unintended results – such as creating infinite loops, or otherwise failing to return – that could needlessly add to your throughput burden.

Usage

When added to a Pipeline, the Code Function offers the following configuration options:

Filter: JavaScript filter expression 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.

Final: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

Code: The mini-editor where you type your JavaScript code.

Advanced Settings

Maximum number of iterations: The maximum number of iterations per instance of this Code Function. Defaults to `5,000`; highest allowed value is `10000`.

Error log sample rate: Specifies the rate at which this Code Function logs errors. For example, a value of `1` logs every error; a value of `10` logs every tenth error. The highest allowed value is `5000`. Defaults to `1`.

Use unique log channel: When enabled, Cribl Edge sends logs from this function to a unique channel in the form `func:code:${pipelineName}:${functionIndex}`. Toggle off to use the generic `func:code` log channel instead.

Notes and Examples

Functions (including the Code Function) always use the special variable `__e` to access the `(context)` event inside JavaScript expressions.

Possibly the simplest Code Function creates a new field and then assigns it a value:

A Basic Code Function

```
__e['foo'] = 'Hello, Goats!'
```

For more ambitious implementations, see Code Function Examples.

JavaScript Support

Cribl Edge supports the ECMAScript® 2015 Language Specification.

With some exceptions, the Code Function supports the options described in the following MDN JavaScript Guide topics:

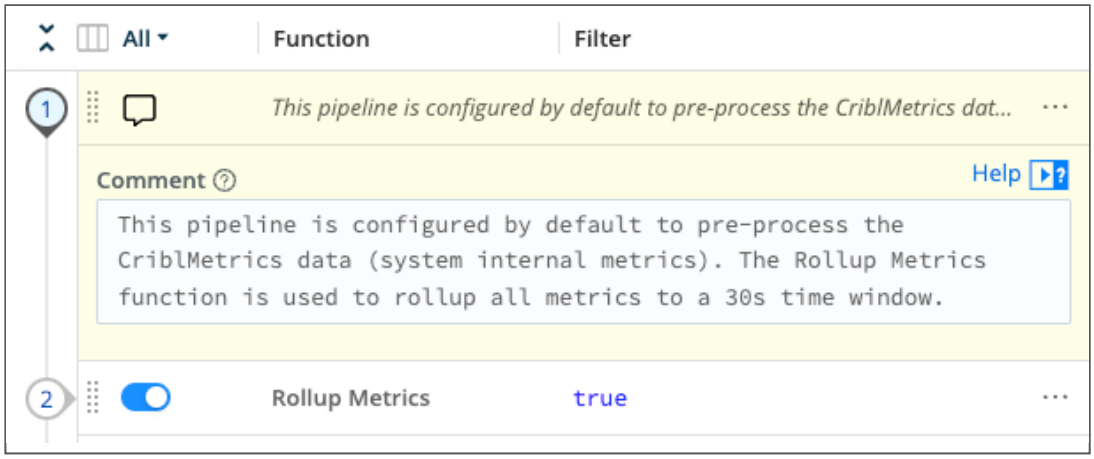
- Expressions and Operators
- Global variables
- Control flow and error handling
- Loops and iteration
- Functions
- Numbers and dates
- Text formatting
- Regular Expressions
- Indexed collections
- Keyed collections
- Working with Objects

13.7. 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.

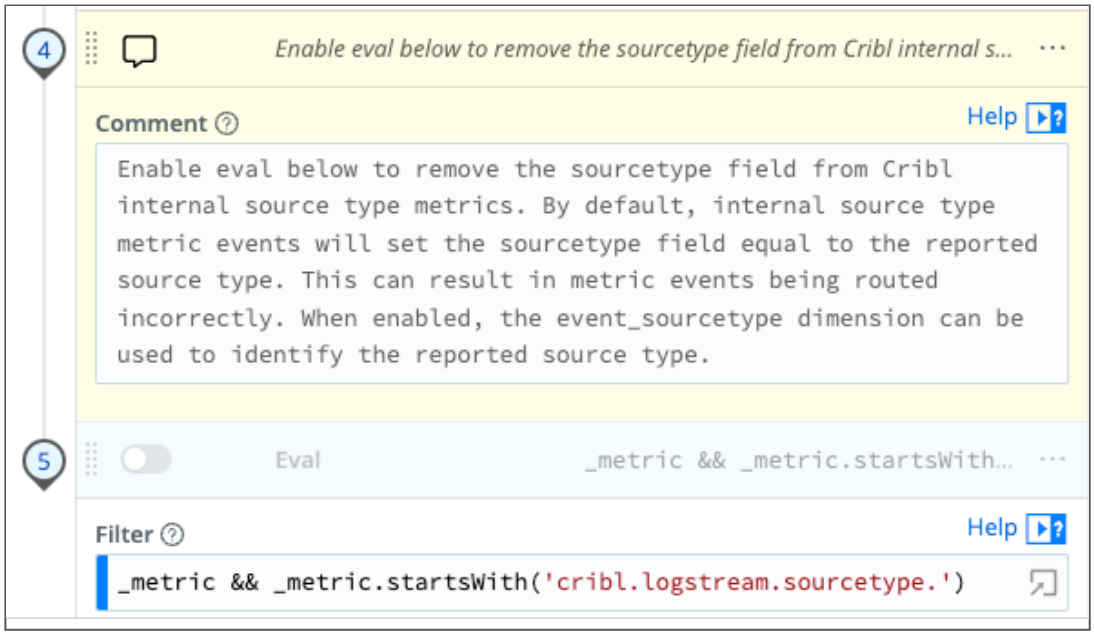
Usage

Comment: Add your comment as plain text in this field.



Examples

This comment labels the Pipeline's next function:



13.8. 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 (JS) 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: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

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 Field(s) 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 to `Yes` if you want Cribl Edge to resolve DNS short names using the search or domain directive from `/etc/resolv.conf`. Defaults to `No`. 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 to `Yes` 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(s) 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.

Maximum cache size: 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`.

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.

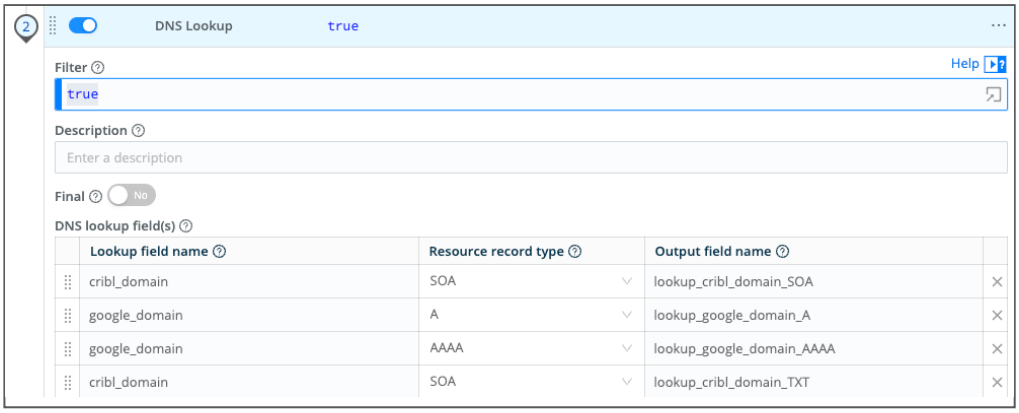
See Monitoring for a general explanation of logging in Cribl Edge.

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.

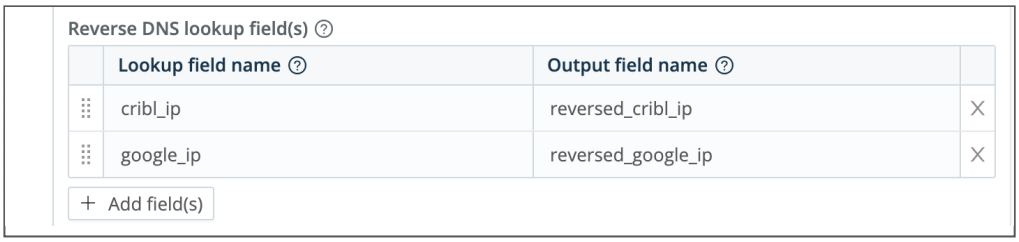
DNS Lookup: Eval Function

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.

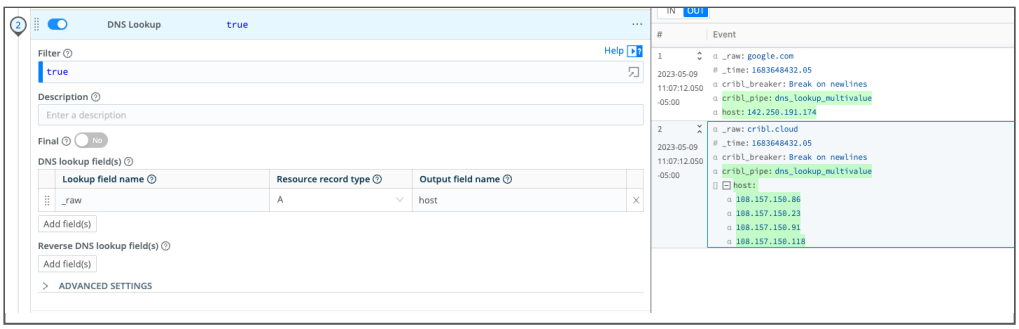


DNS Lookup: reverse lookups

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.



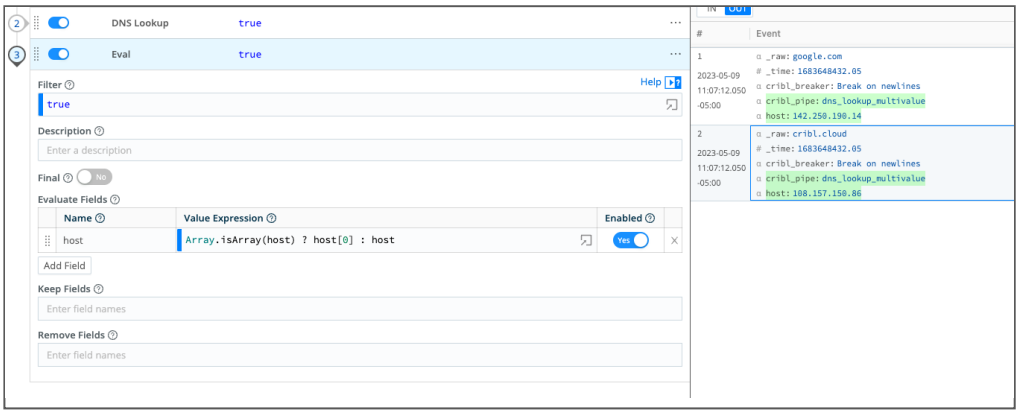
DNS lookup with multi-value result

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;
```

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
```

Here, instead of picking the first element, the code uses `Math.floor(Math.random() * host.length)` to randomly pick an element from the array.

13.9. Drop

The Drop Function drops (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 (JS) 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: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

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'`

13.10. 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 (JS) 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: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

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`.
- **Max sampling rate**. 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

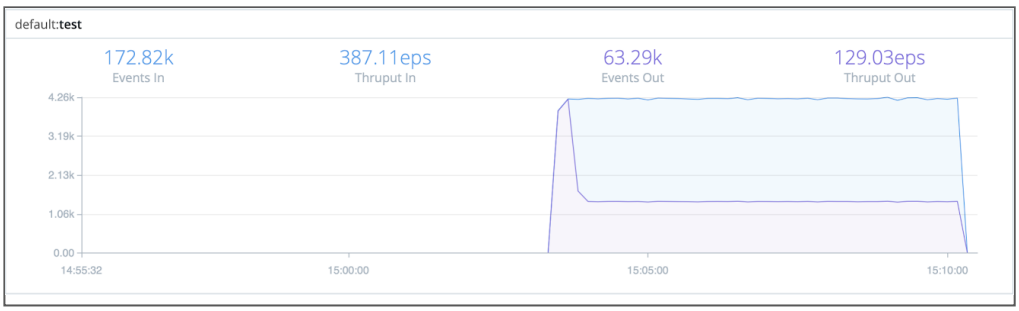
Here's an example that illustrates the effectiveness of using the Square Root sample mode.

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.

13.11. Eval

The Eval Function adds or removes fields from events. (In Splunk, these are index-time fields.)

Usage

Filter: Filter expression (JS) 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: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

Evaluate fields: Table of event fields to evaluate and add/set. Click **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 details, see Cribl Expression Syntax.)
- **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. Supports wildcards (*) and nested addressing. Takes precedence over **Remove fields** (below). To reference a parent object and all children, you must use the (*) wildcard. For example, if `_raw` is converted to an object then use `_raw*` to refer to itself and all children.

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*`.

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=='fail' ? 'blocked' : action`

Scenario C: Create a multivalued field called `myTags`. (i.e., 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>
```

To Append:

```
parent =  Object.assign(parent, { child2: child2Value })
```

To Create a New Field:

```
parent2 =  Object.assign(parent, { child2: child2Value, child3: child3Value })
```

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 JS expression. In this case, content parsed from `_raw` is added at the top level of the event.

13.12. Event Breaker

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 (JS) 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: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

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 `Yes`.

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.

13.13. Flatten

The Flatten Function flattens fields out of a nested structure. It pulls up nested key-value pairs (fields) 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 (JS) 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: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

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
        }]
}
```

Under **Select Event Breaker**, choose **ndjson** (newline-delimited JSON), and click **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,
}
```

Using the Flatten Function's default settings, we successfully create top-level fields from the nested JSON structure, as expected.

See Also

- Reducing Windows XML Events, which demonstrates flattening within `_raw` events ingested via Windows XML.

13.14. GeoIP

The GeoIP Function enriches events with geographic fields, given an IP address. It works with MaxMind's GeoIP binary database.

Prerequisite

You need to host the `.mmdb` database file from MaxMind. The following steps cover this process at a high level. They link to our Managing Large Lookups 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 Reducing Deploy Traffic.
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 of the database file.

Usage

Filter: Filter expression (JS) 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: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

GeoIP file (.mmdb): Path to a MaxMind 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 MaxMind 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 click this combo 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.

13.15. Grok

The Grok Function extracts structured fields from unstructured log data, using modular regex patterns.

Usage

Filter: Filter expression (JS) 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: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

Pattern: Grok pattern to extract fields. Cick the Expand button at right to open a preview/valdiation modal. Syntax supported: `%{PATTERN_NAME:FIELD_NAME}`.

Click **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"}
```

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",
}
```

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}`. E.g.: `%{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

13.16. JSON Unroll

The JSON Unroll Function accepts a JSON object string `_raw` field, unrolls/explodes an **array** of **objects** therein into individual events, while also inheriting top level fields. See example(s). Cribl highly recommends not using this JSON Unroll function for certain types of data. Instead, perform the unrolling using an event breaker for those inputs which support configuring an event breaker. Specifying the event breaker type **JSON Array** and toggling the **JSON Extract Fields** option to **Yes** will accomplish the same unrolling but much more efficiently. This is recommended, for example, for CloudTrail and Office635 events, which are collected as JSON arrays.

Usage

Filter: Filter expression (JS) 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: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

Path: Path to array to unroll, e.g., `foo.0.bar`.

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(s)

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" ] }
   ]
}
```

Settings:

Path: `allCars` **New Name**: `cars`

Output Events:

Resulting Events


```
Event 1:
{"_raw":"{"date":"9/25/18 9:10:13.000 PM","name":"Amrit","age":42,"cars":{"name":"Fo

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":"Fi

Event 4:
{"_raw":"{"date":"9/25/18 9:10:13.000 PM","name":"Amrit","age":42,"cars":{"name":"Bl
```


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**

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.

13.17. 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 (JS) 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: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

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 Library, or specify the path. You can reference environment variables via `$`, e.g.: `$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).

Match mode: Defines the format of the lookup file, and indicates the matching logic that will be performed. Defaults to `Exact`.

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.

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

Reload period (sec): To periodically check the underlying file for mod-time changes, and reload the file if necessary, change the default `-1` value (disabled) to a positive integer representing the check interval in seconds.

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. (E.g., 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.

Ignore case: Ignore case when performing **Match mode: Exact** lookups. Defaults to `No`.

Add to raw event: Whether to append the looked-up values to the `_raw` field, as key=value pairs. Defaults to `No`.

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
```

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,2049,2018/0
{"_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,2049,2018/0
  "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"
  }
```

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
```

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/(
  "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/(
  "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/(
  "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/(
  "destination_ip": "10.0.0.102",
  "location": [
    "San Francisco",
    "California",
    "US",
  ]}
```

More Examples and Scenarios

For usage examples, see these Better Practices topics and other resources:

- Ingest-time Lookups: Enriching data in motion for fast access and on-demand accuracy.
- Lookups and Regex Magic: Combining the Regex Extract, Lookup, and Eval Functions to enrich events with region, index, and sourcetype data.
- Lookups as Filters for Masks: Using lookups to optimize multiple Mask Functions in a single Pipeline.
- Managing Large Lookups to optimize file locations for large lookup files.

- Updating Lookups in Multiple Packs: Efficiently manage frequent updates to a large lookup file.
- Updating Lookups with a REST API: Using API calls to refresh a lookup file, and commit and deploy it to Worker Groups.
- Lookup Examples Pack: Examples of using the Lookup Function and C.Lookup Expressions.

See Also

- C.Lookup Expressions: Lookup methods that you can use in JavaScript expressions.
- 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.

13.18. Mask

The Mask Function masks, or replaces, patterns in events. This is especially useful for redacting PII (personally identifiable information) and other sensitive data.

Usage

Filter: Filter expression (JS) 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: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

Masking Rules: This table defines pairs of patterns to match and replace. Defaults to empty. Click **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, e.g.: `/foo(bar)/g`
- **Replace Expression**: A JavaScript expression or literal to replace all matching content. Capture groups can be referenced with `g` and the group number – e.g., `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. E.g., `!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).

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.

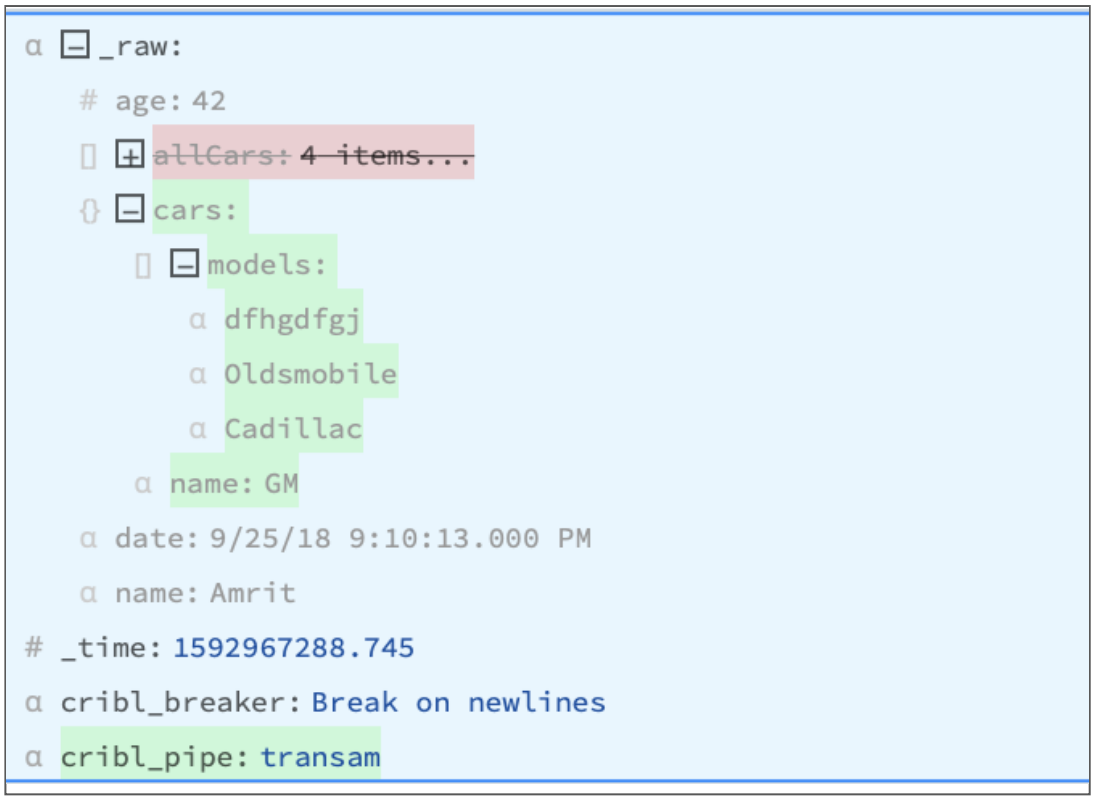
For example, suppose a **Replace expression** that generates a random integer is applied to a field named `bytes` and produces the number `8403`. Cribl Edge will output an event where the value of the `bytes` field is the **string** `8403`.

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

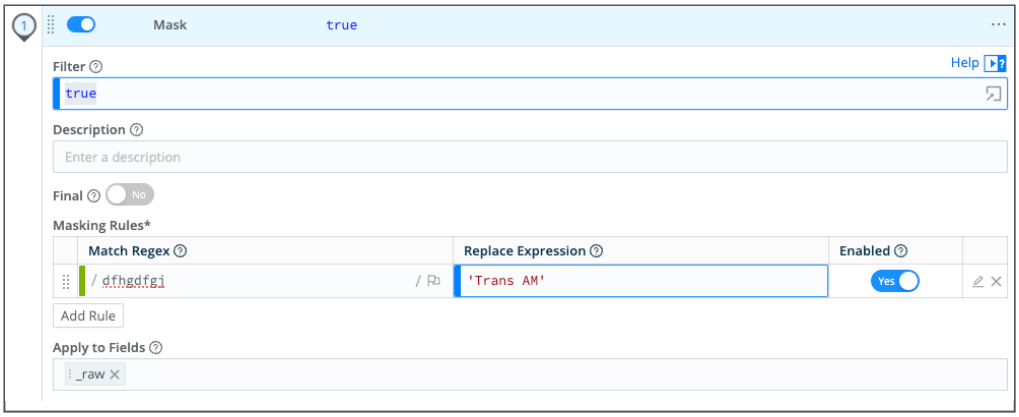
Here, we'll simply search for the string `dfhgdfgj`, and replace that value (if found) with `Trans AM`. This will help close America's muscle-car gap:



Event before masking

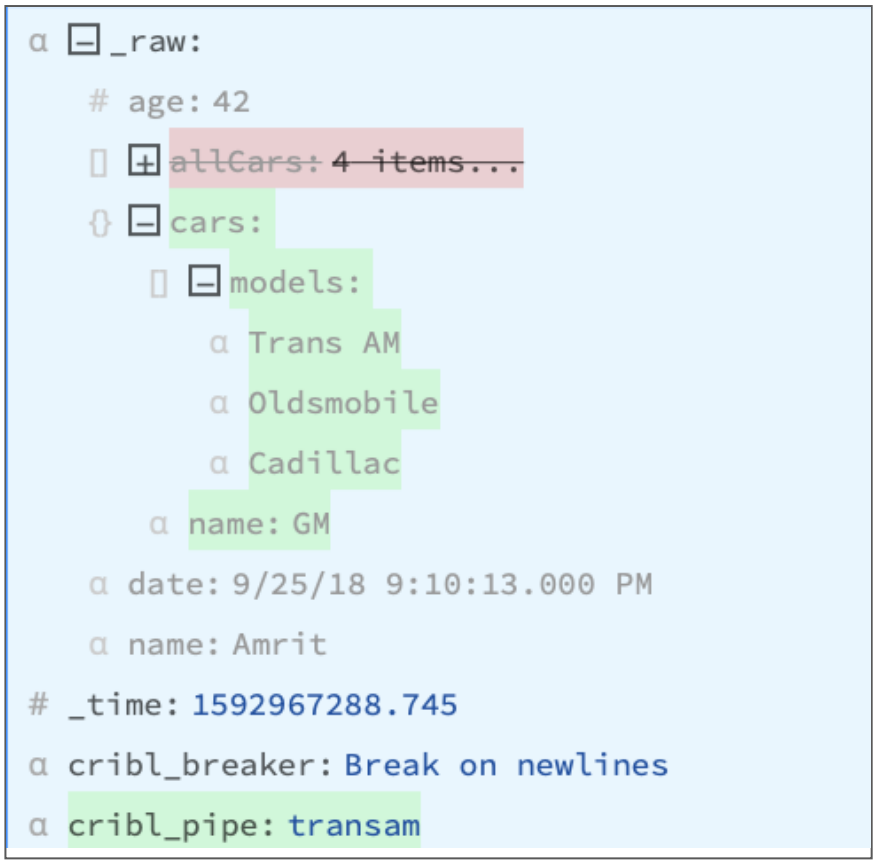
Configure the Mask Function > Masking Rules as follows:

Match Regex: `dfhgdfgj` Replace Expression: `Trans AM`



Mask Function configuration

Result: Vroom vroom!



Event after masking

Example 2: Mask Sensitive Data

Assume that you're ingesting data whose `_raw` fields contain unredacted Social Security numbers in the Key=Value pattern `social=#########`.

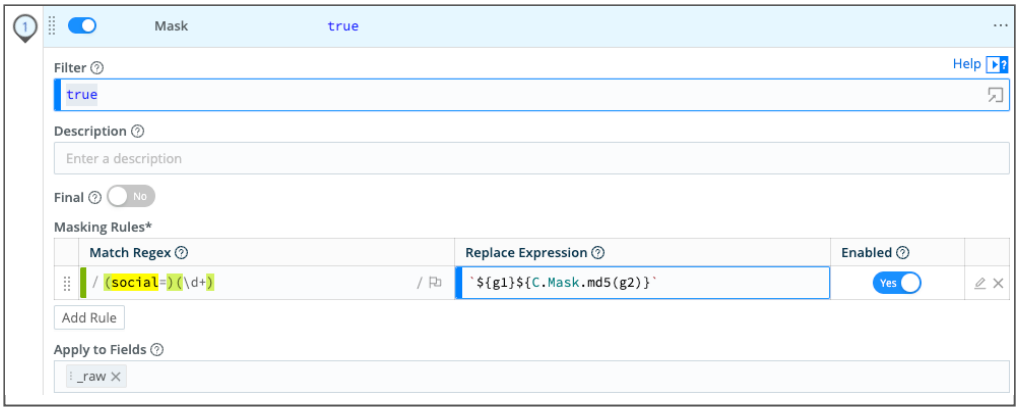
Event with unredacted SSNs

You can use a Mask Function to run an md5 hash of the `social` keys' numeric values, replacing the original values with the hashed values. Configure the Masking Rules as follows:

Match Regex: `(social=)(\d+)` Replace Expression: `` `${g1}${C.Mask.md5(g2)}` ``

In the first example everything in the Match regex field was replaced by the Replace Expression. However if that isn't desired then you can use capture groups in the Match Regex to define individual string components for manipulation or, alternatively, use string literals in the Replace expression for retaining any static text. Any content matching the Match Regex that is not inserted into the Replace expression will not be retained.

In this example, `social=` is assigned to capture group g1 for later reference. The value of `social=` will be hashed by referencing it as g2 in the md5 function. If we didn't make `social=` its own capture group (or specified `social=` as a literal in the Replace Expression) then we cannot reference it using g1 in the Replace expression, the value of `social=` would instead be assigned to g1, and the entire `social=#########` string would be replaced with a hash of the social security number, which probably isn't desired because no one would know the value being hashed without a field name preceding it.



Mask Function configuration

Result: The sensitive values are replaced by their md5 hashes.

In scenarios where you need to send unmodified values to certain Destinations (such as archival stores), you can narrow the Mask Function's scope by setting the associated Route's **Output** field.

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
```

To match the IP address, we'll define three capture groups. Set **Match Regex** to:

```
/(IpAddress=)((?:\d{1,3}\.){3}\d{1,3})(\s)/
```

In the **Replace Expression**, we'll reference the `source_ip` field by prepending `event` to it, like this:

```
${g1}${event.source_ip}${g3}
```

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
```

See Also

For further usage examples, see:

- Masking and Obfuscation illustrates random, repeat, string, and hash masking.
- The Cribl Knowledge Pack provides sample Pipelines that demonstrate using Mask to both encode and decode data, convert hex values to numbers, and create md5 hashes of values using multiple wildcards.

13.19. Numerify

The Numerify Function converts event fields that are numbers to type `number`.

Usage

Filter: Filter expression (JS) 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: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

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 Wildcard Lists.

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.

13.20. 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 (JS) 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-log events: Whether or not to drop any non-log events. Defaults to `No`.

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
}
```


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 |
            "attributes": [],
            "dropped_attributes_count": 0,
            "severity_number": 9,
            "severity_text": "INFO"
        }],
    }],
    "_time": 1717010325.183
}
```


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
}
```


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
}
```


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": []
            },
            "log_records":[{
                "time_unix_nano": 1717016223073301200,
                "observed_time_unix_nano": 1717016223073302800,
                "body": { "string_value": "[KafkaRaftServer nodeId=1] Kafka Server s
                "attributes": [],
                "dropped_attributes_count": 0,
                "severity_number": 9,
                "severity_text": "INFO",
                "_time": 1717016223.0733013
            }],
        },
        {
            "scope": {
                "name": "kafka.server.KafkaBoatServer",
                "version": "unknown",
                "attributes": []
            },
            "log_records":[{
                "time_unix_nano": 1717016223073401200,
                "observed_time_unix_nano": 1717016223073402800,
                "body": { "string_value": "[KafkaBoatServer nodeId=1] Boat Server st
                "attributes": [],
                "dropped_attributes_count": 0,
                "severity_number": 9,
                "severity_text": "INFO",
```

```
                "_time": 1717016224.0733013
            }],
        }
    ],
    "_time": 1717010325.183
}
```

13.21. 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 (JS) 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`.

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`. See

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 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 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"
}
```

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"
  }
]
```

```
      }
    ]
  }
```

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_webl
                                }
                            },
                            {
                                "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"
}
```

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"
}
```

```
{
    "_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_webl
                        }
                    },
                    {
                        "key": "source",
                        "value":
                        {
                            "string_value": "cribl"
                        }
                    }
                ]
            }
        ]
    },
    "__type": "gauge",
    "description": "This is a metric description"
},
{
    "name": "request_duration_summary",
    "summary":
    {
```

```
                "data_points":
                [
                    {
                        "as_int": 0,
                        "count": 0,
                        "sum": 0,
                        "quantile_values":
                        {
                            "0.5": 0,
                            "0.9": 0
                        },
                        "time_unix_nano": 1701793930000000000,
                        "start_time_unix_nano": 1701793930000000000,
                        "attributes":
                        [
                            {
                                "key": "cribl_host",
                                "value":
                                {
                                    "string_value": "P4MD6R-Pv8x"
                                }
                            },
                            {
                                "key": "cribl_wp",
                                "value":
                                {
                                    "string_value": "w0"
                                }
                            }
                        ]
                    }
                ]
            },
            "__type": "summary",
            "description": "This is an event description"
        }
    ],
    "instrumentation_library":
    {
        "name": "io.opentelemetry.sdk.trace",
        "version": "1.33.0-alpha"
    }
}
```

```
    ],
    "__outLen": 1266,
    "__bufToken": "otlp-metrics-batcher1"
}
```

13.22. 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 (JS) 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"
    }
  }]
}
```


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"
    }
  }]
}
```


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"
              }
            },
            {
              "key": "http.url",
              "value": {
                "string_value": "/convert"
              }
            }
          ]
        }
      ]
    }
  ]
}
```

13.23. Parser

The Parser Function can be used to extract fields out of events, or to reserialize (rewrite) events with a subset of fields. Reserialization will maintain 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.

Parser cannot remove fields that it did not create. A subsequent Eval Function can do so.

Usage

Filter: Filter expression (JS) 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: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

Operation mode: **Extract** will create new fields. **Reserialize** will extract, filter fields, and then reserialize.

Type: Parser/Formatter type to use. Options:

- CSV – selecting this type requires you to escape certain characters.
- Extended Log File Format (ELFF)
- Common Log Format (CLF)
- K=Value Pairs – selecting this type requires you to escape certain characters.
- JSON Object
- Delimited Values – selecting this type expands the UI.

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
- Regex Extract

Library: Select an option from the Parsers Library. 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 add extracted and serialized fields. 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 to `Yes` 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. Supports wildcards (`*`). Takes precedence over **Fields to remove**. Supports nested addressing. See details below.

Fields to remove: List of fields to remove. Supports wildcards (`*`). Cannot remove fields matching **Fields to keep**. Supports nested addressing. See details below.

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**.

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. E.g., `!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

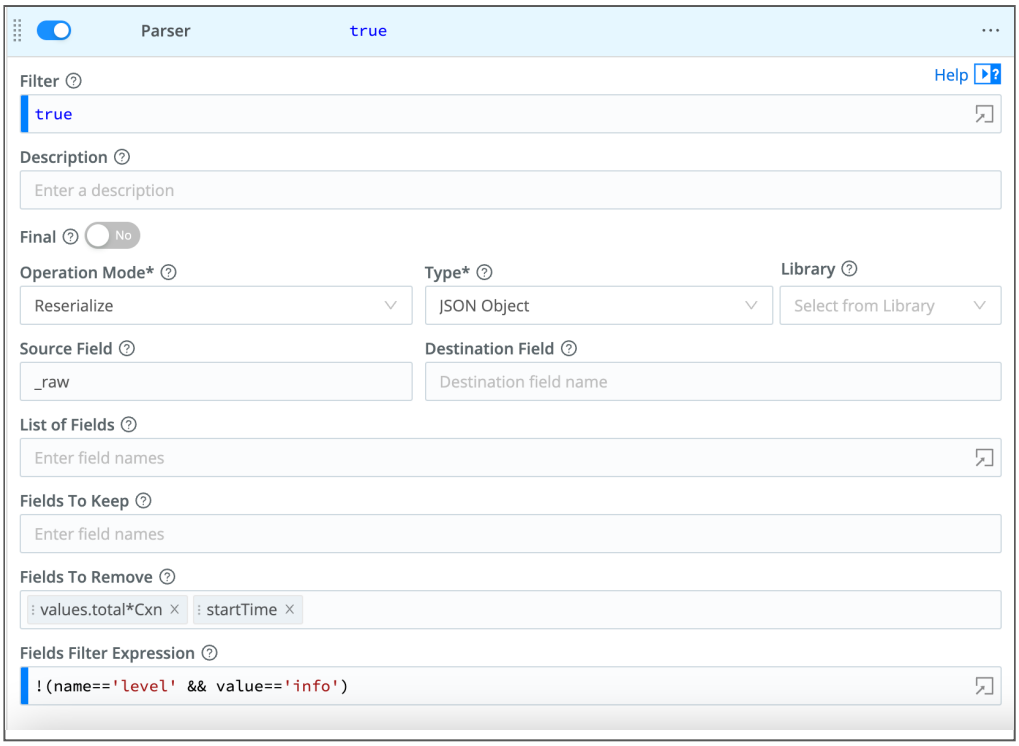
Examples

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 (or import this Pipeline: https://raw.githubusercontent.com/weeb-cribl/cribl-samples/master/parser/functions/parser/parser_1.json).



Parser Function initial configuration

First, set the **Type** to `Key=Value Pairs`.

Scenario A

Keep fields `a, b, c`. Drop the rest.

Expected result: `a, b, c`

- Fields to Keep: `a, b, c`
- Fields to Remove: `*`
- 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 clicking 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`

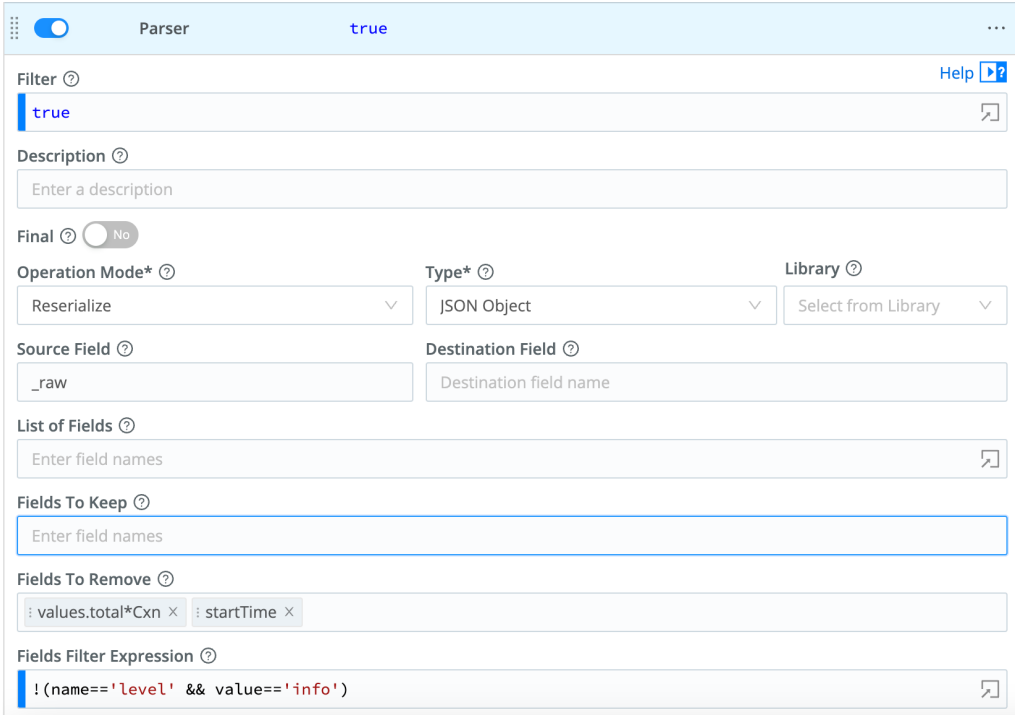
The `index` refers to the location of a field in the array of all fields extracted by **this** Parser. It is zero-based. In the case above, `g2` and `g3` have `index` values of `7` and `8`, respectively.

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:

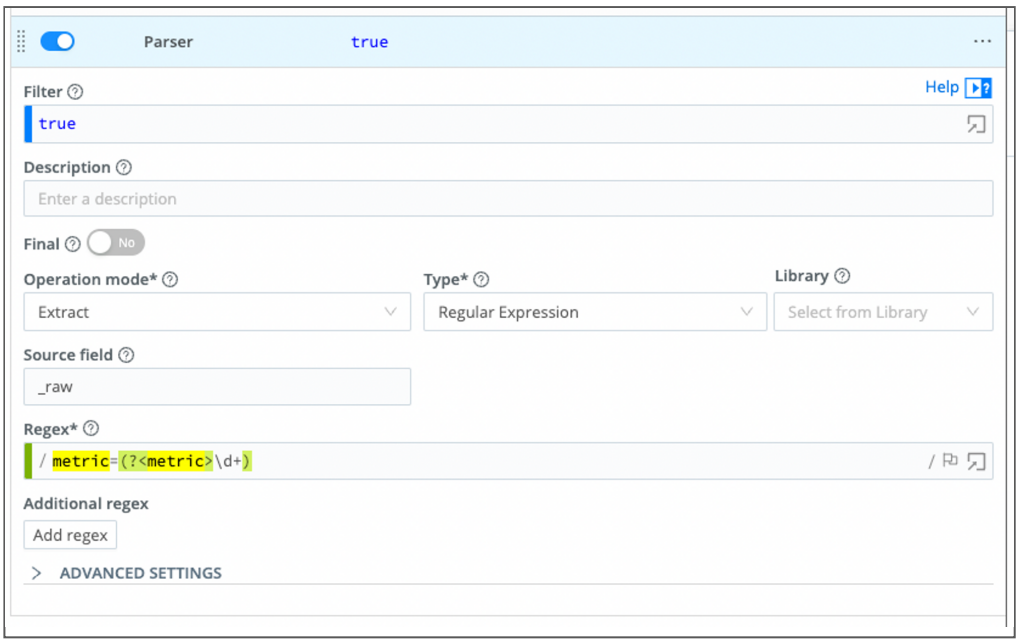
- Operation mode: `Serialize`
- Source field: [empty]
- Destination field: `bar`
- Type: `JSON`
- List of fields: [empty]
- Fields to keep: `a, b, c, d`

Example 4

Assume we have a simple event that looks like this:

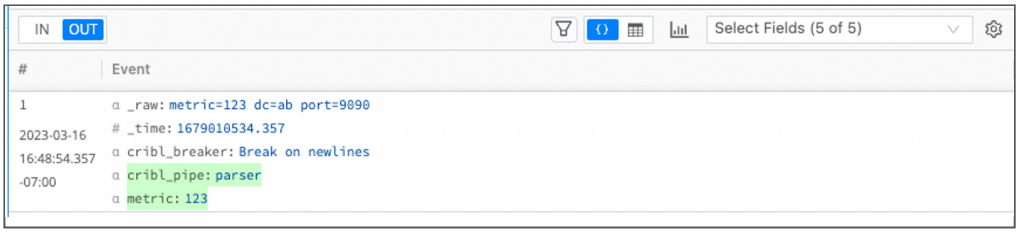
```
metric=123 dc=ab port=9090
```

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

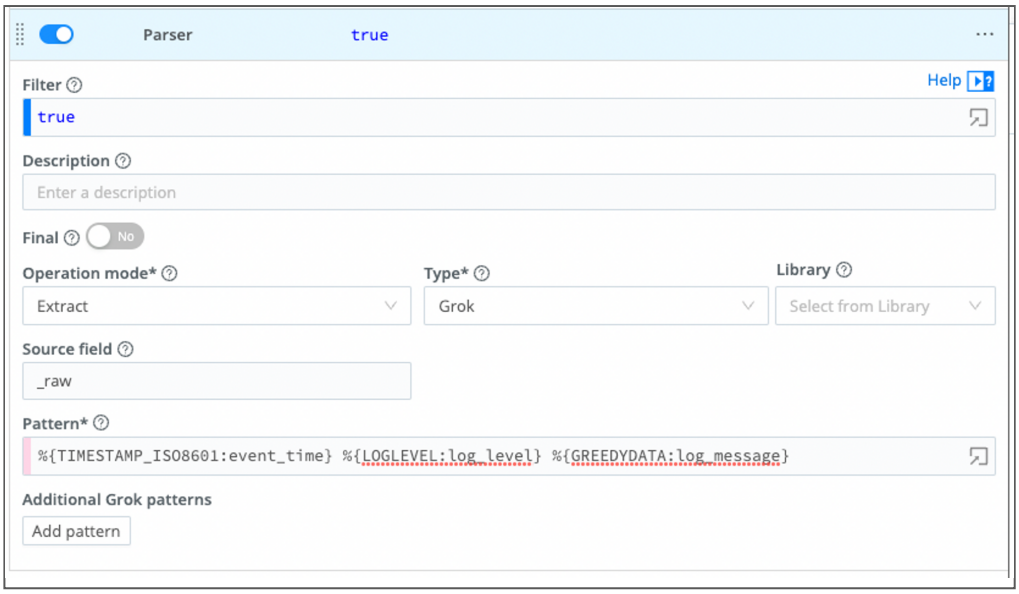
Assume we have an event that looks like this:

```
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

13.24. Publish Metrics

The Publish Metrics Function extracts, formats, and outputs metrics from events.

Usage

Filter: Filter expression (JS) 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: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

Overwrite: If set to `Yes`, overwrite previous metric specs. Otherwise, append. Defaults to `No`.

Metrics

Add Metrics: List of metrics to extract from the event and format. Destinations can pass the formatted metrics to a metrics aggregation platform. Click **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.

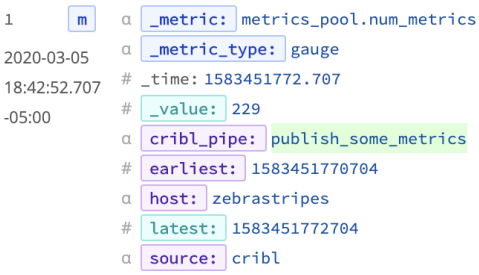
Overwrite: If set to `Yes`, overwrite previous metric specs. Otherwise, append. Defaults to `No`.

The **Add Dimensions** and **Remove Dimensions** fields support wildcards and negated terms. When you use negated terms, the list is order-sensitive. E.g., `!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 – i.e., 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,
}
```


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
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`.

13.25. Redis

The Redis Function interacts with Redis stores, setting and getting key-hash and key-value combinations. Redis' in-memory caching of these key pairs enables 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

When using Redis Functions, you must consider how to avoid creating large numbers of TCP connections to the Redis server (which we'll call "Redis connections"). To manage Redis connections, you start by calculating how many Redis connections your Redis Functions, other functions, and Pipelines will create using default settings. Then, to keep the overall number of connections reasonable, you configure Cribl Edge to **reuse** Redis connections.

What causes the number of Redis connections to grow large? By default, 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; and, using a Redis Function in a Destination's post-processing Pipeline.

How can you control the number of Redis connections created? You use the **Reuse Redis connections** control. Then, when a new Redis connection is needed, **and** its connection properties are identical to those of an existing Redis connection, Cribl Edge will try to reuse the existing connection instead of creating a new one. The example below illustrates how powerfully this can affect the demands Cribl Edge places on a Redis server.

What are the connection properties that must be identical for reuse to be possible? For the different kinds of Redis instances, the relevant connection properties are, respectively:

- 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.

In managing Redis connections, the factors you must account for include: what kind of Redis instance you're connecting to (Standalone, Cluster, or Redis Sentinel); the size of the Redis server; the number of connections the Redis server is configured to allow; the number of other clients using the same Redis server; and so on.

Where to configure **Reuse Redis connections** in the UI depends on how you have deployed Cribl Edge:

- Distributed deployment: Navigate to **Limits** at the Worker Group level.
- Single-instance deployment: Navigate to **Limits** at the global (**General Settings**) level.

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 **Max connections** 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 (JS) 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: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

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. Click the icon at right to open a validation modal.

Args: A JavaScript expression to compute arguments to the operation. Click 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 `Yes`.

Server name (SNI): Server name for the SNI (Server Name Indication) TLS extension. This must be a host name, not an IP address.

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.

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

Max blocking time: 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 to `Yes` 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 to `Yes` 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.

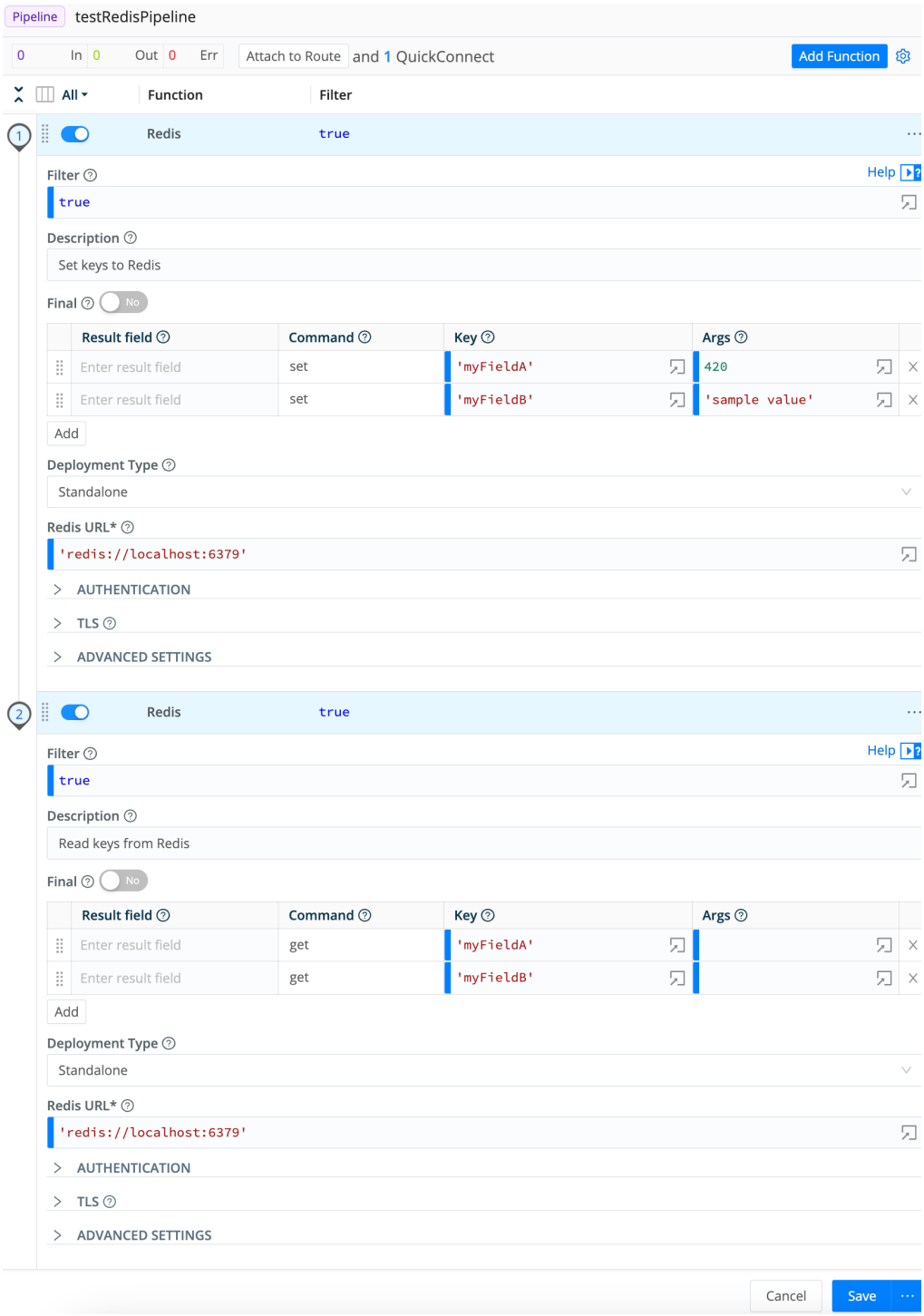
You should select whichever of these mechanisms best suits your environment. Does it make more sense to impose the heavier burden on the Redis server (as in **Default** mode) or on Cribl Edge (as in **Broadcast** mode)?

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

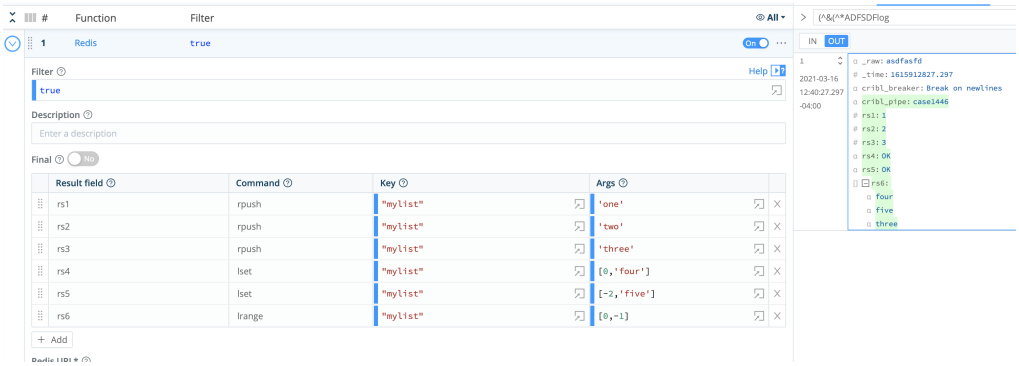
Description: `Read keys from Redis`

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

```
{
  "id": "redis-multiple-args",
  "conf": {
    "output": "default",
    "groups": {},
    "asyncFuncTimeout": 1000,
    "functions": [
      {
        "filter": "true",
        "conf": {
          "commands": [
            {
              "outField": "rs1",
              "command": "rpush",
              "keyExpr": "\"mylist\"",
              "argsExpr": "'one'"
            },
            {
              "outField": "rs2",
              "command": "rpush",
              "keyExpr": "\"mylist\"",
              "argsExpr": "'two'"
            },
            {
              "outField": "rs3",
              "command": "rpush",
              "keyExpr": "\"mylist\"",
              "argsExpr": "'three'"
            },
            {
              "command": "lset",
              "keyExpr": "\"mylist\"",
              "argsExpr": "[0,'four']",
              "outField": "rs4"
            },
            {
              "outField": "rs5",
              "command": "lset",
              "keyExpr": "\"mylist\"",
              "argsExpr": "[-2,'five']"
            },
            {
```

```
            "outField": "rs6",
            "command": "lrange",
            "keyExpr": "\"mylist\"",
            "argsExpr": "[0,-1]"
          }
        ],
        "maxBlockSecs": 60,
        "url": "redis://<your-credentials-here>"
      },
      "id": "redis",
      "disabled": false,
      "description": "Push arrays of multiple arguments to Redis"
    }
  ]
 }
}
```

13.26. Regex Extract

The Regex Extract Function extracts fields using named capture groups. (In Splunk, these will be index-time fields). Fields that start with `__` (double underscore) are special in Cribl Edge. They are ephemeral: they can be used by any Function downstream, but will be serialized only to Cribl internal Destinations.

Usage

Filter: Filter expression (JS) 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: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

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. See Examples below.

Additional regex: Click **Add Regex** to chain extra regex conditions.

Source field: Field on which to perform regex field extraction. Nested addressing is supported. Defaults to `_raw`.

Overwrite existing fields: Whether to overwrite existing event fields with extracted values. If set to `No` (the default), existing fields will be converted to an array. If toggled to `Yes`, Regex Extract will create array fields if applied multiple times, or if fields exist. For example, if `src_ip` is extracted in an input Pipeline where it is assigned a value of `10.1.2.2`, and is also 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 `_NAME_n` and `_VALUE_n` capturing groups are used. 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
```

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-(
```

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, click **Paste** and insert the following sample:

Sample Data

```
<134>1 2020-12-22T17:06:08Z CORP_INT_NLB CheckPoint 18160 - [action:"Accept"; conn_d
```

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.

13.27. Regex Filter

The Regex Filter Function filters out events based on regex matches.

Usage

Filter: Filter expression (JS) 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: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

Regex: Regex to test against. Defaults to empty.

Additional regex: Click **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.

13.28. Rename

The Rename Function is designed to change fields' names or reformat their names (e.g., by 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 (JS) 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: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

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. Click **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 other fields' values via `event.<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
}
```

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
}
```

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')`

See Also {see-also}

- Reducing Windows XML Events, which demonstrates using Rename to remove unnecessary prefixes from field names, reducing events' size.

13.29. Rollup Metrics

The Rollup Metrics Function merges/rolls up 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 (JS) 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: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

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 (e.g., `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 – e.g. `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 – e.g., `10s`. In the **Dimensions** field, you would remove the default `*` wildcard, and would specify only the dimensions you want to keep – e.g.: `host, data_center`.

13.30. 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 (JS) 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: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

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.

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, click **Simple** beside the new file's name. If you then click 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 onbaording 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

13.31. Serialize

Use the Serialize Function to serialize an event's content into a predefined format.

Usage

Filter: Filter expression (JS) 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: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

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 to `Yes` 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: You must specify individual fields with the `CSV`, `ELFF`, `CLF`, and `Delimited values` Types. All other formats support wildcard field lists. With these formats, the default entry here will specify some excluded fields (indicated by `!`), followed by a `*` wildcard to serialize all remaining fields.

All field names or wildcards that you want to exclude from serialization must precede all field names or wildcards that you want to include.

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 click **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`

13.32. SNMP Trap Serialize

Use the SNMP Trap Serialize Function to serialize 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 (JS) 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: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

Enforce required fields: When toggled to `Yes` (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 to `Yes` (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 to `No`, the event will pass through without `__snmpRaw` and will include a `snmpSerializeErrors` field for troubleshooting.

Regardless of whether the toggle is set to `Yes` or `No`, 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 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';`

13.33. Suppress

The Suppress Function suppresses events over a time period, based on evaluating 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 (JS) 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: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

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 `Yes`, meaning drop.

Advanced Settings

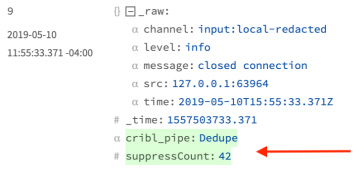
Maximum cache size: The maximum number of keys that can be cached before idle entries are removed. Before changing the default `50000`, contact Cribl Support to understand the implications.

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 **Maximum cache size**. 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: `No`

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 Filters.

13.34. 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 (JS) 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: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

Command: Command to execute and receive events (via `stdin`) – one JSON-formatted event per line.

Args: Click **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 `Yes`.

Environment variables: Environment variables to set or overwrite. Click **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 clicked **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.

13.35. Unroll

The Unroll Function accepts an array field – or an expression to evaluate an array field – and breaks/unrolls the array into individual events.

Usage

Filter: Filter expression (JS) 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: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

Source field expression: Field in which to find/calculate the array to unroll. E.g.: `_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/systemd
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]
```

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/systemd

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]
```

13.36. 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 (JS) 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: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

Unroll elements regex: Path to the array to unroll. E.g.: `^root\.child\.ElementToUnroll$`

Copy elements regex: Regex matching elements to copy into each unrolled event. E.g.: `^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>
```

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>
  <branchLocation>US</branchLocation>
  <state>CA</state>
  <city>San Francisco</city>
</Child>
```

See Also

- Reducing Windows XML Events, which demonstrates using several Cribl Stream Functions to parse WindowsXML events and significantly reduce their volume.

13.37. 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 (JS) 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: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

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: If set to `No` (the default), overrides the **Final** setting, and suppresses output to downstream Functions' Destinations. Toggle to `Yes` 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: On the default `No` setting, suppresses output to downstream Functions' Destinations. (This overrides the **Final** setting.) Toggle to `Yes` 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="

# TYPE perf_192_168_1_248_pkts_rcvd counter
metric_192_168_1_248_pkts_rcvd {destination_ip="160.177.222.249",inbound_interface="
```

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.

13.38. 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 (JS) 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: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

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
α _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
α app_proto: HTTPS
α client_app: SSL client
α client_version:
α connection_id: 21378
α cribl_breaker: Break on newlines
α cribl_pipe: rev_dns
α dest_autonomous_system: 0
α dest_bytes: 3746
α dest_host: dns.google
α dest_ip: 8.8.8.8

13.39. 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 (JS) 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: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

Field name: Name of field in which to save the timestamp. (If empty, timestamp will not be saved to a field.)

Example

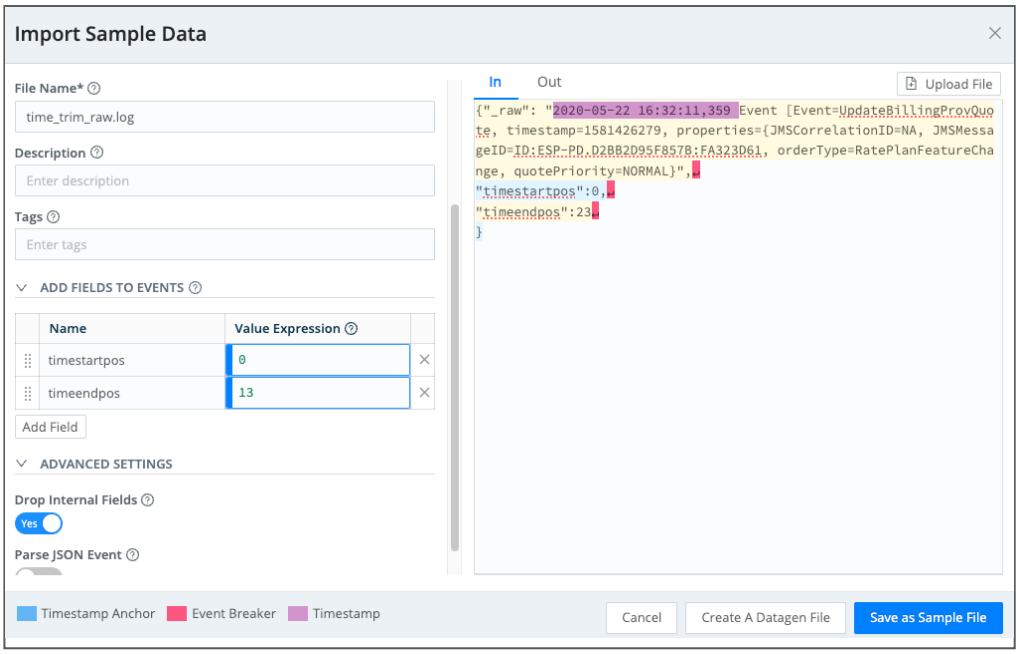
Remove the timestamp pattern (indicated by `timestartpos` and `timeendpos`) from `_raw`, and stash it in a field called `time_field`.

Field name: `time_field`

Example event before:

```
{"_raw": "2020-05-22 16:32:11,359 Event [Event=UpdateBillingProvQuote, timestamp=158
"timestartpos":0,
"timeendpos":23
}
```

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:

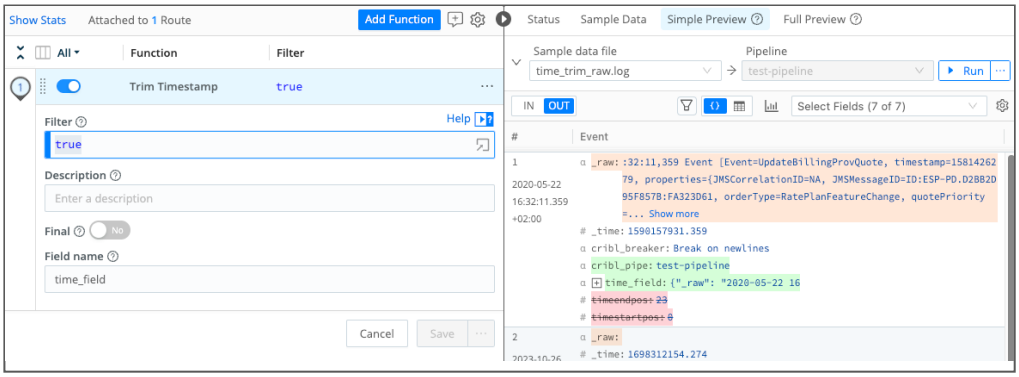


Example event setup

Example event after:

```
{"_raw": "Event [Event=UpdateBillingProvQuote, timestamp=1581426279, properties={JMS
"time_field":"2020-05-22 16:32:11,359"
}
```

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

14. Knowledge Libraries

Select **Processing** > **Knowledge** to access Cribl Edge's libraries of default Lookup tables, Event Breaker rulesets, Parsers, Global Variables, Regex expressions, Grok patterns, Schemas, Database Connections, and AppScope config templates. You can customize these libraries by adding your own Knowledge objects.

From the **Processing** > **Knowledge** top nav, you can select:

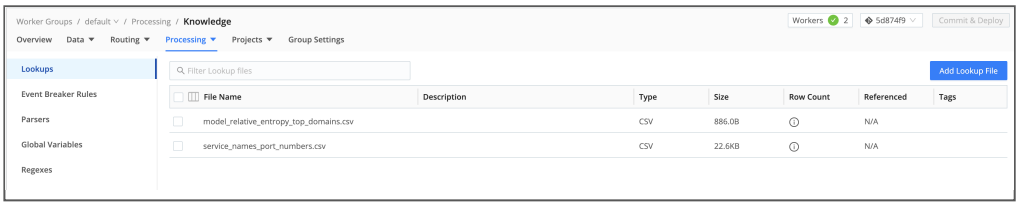
- Lookups
- Event Breakers
- Parsers
- Global Variables
- Regexes
- Grok Patterns
- Schemas (JSON)
- Parquet Schemas
- Database Connections (Cribl Stream only)
- HMAC Functions (Cribl Stream only)
- AppScope Configs

14.1. Lookups Library

What Are Lookups

Lookups are data tables that can be used in Cribl Edge to enrich events as they are processed by the Lookup Function. To access the Lookups library, which provides a management interface for all lookups, navigate to **Products** > **Cribl Edge** > **Fleets**. Select a Fleet, then go to **Processing** > **Knowledge** > **Lookups**.

This library is searchable, and each lookup can be tagged as necessary. There's full support for `.csv` files. Compressed files are supported, but must be in gzip format (`.gz` extension). You can add files in multimedia database (`.mmdb`) binary format, but you cannot edit these binary files through Cribl Edge's UI.



Lookups Library

How Does the Library Work

In single-instance deployments, all files handled by the interface are stored in `$CRIBL_HOME/data/lookups`. In distributed deployments, the storage path on the Leader Node is `$CRIBL_HOME/groups/<groupname>/data/lookups/` for each Worker Group.

For large and/or frequently replicated lookup files, you might want to bypass the Lookups Library UI and instead manually place the files in a different location. This can both reduce deploy traffic and prevent errors with Cribl Edge's default Git integration. For details, see Managing Large Lookups.

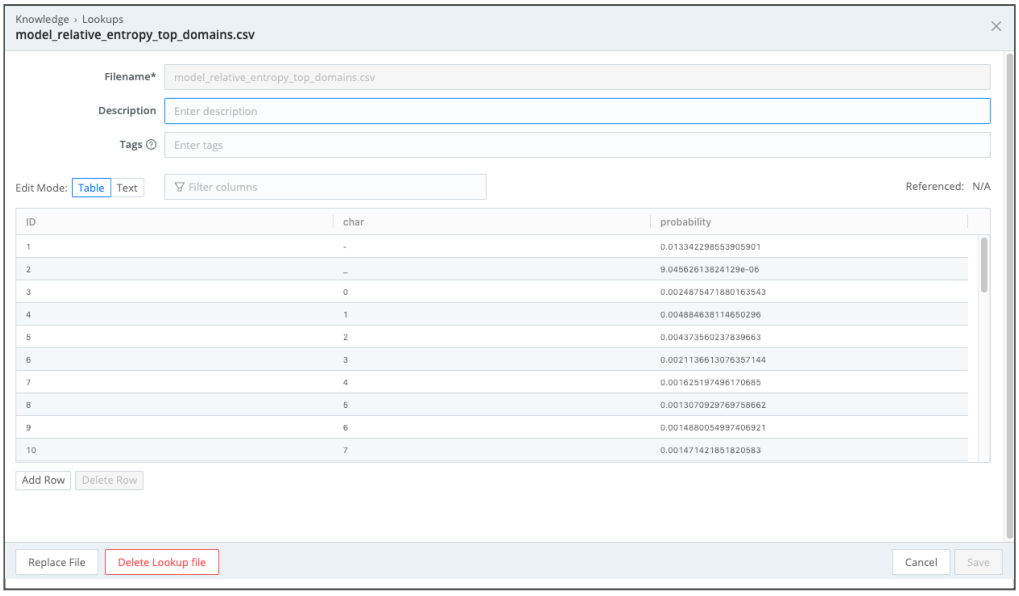
Add Lookup Files

To upload or create a new lookup file, select **New Lookup File**, then select the appropriate option from the drop-down.

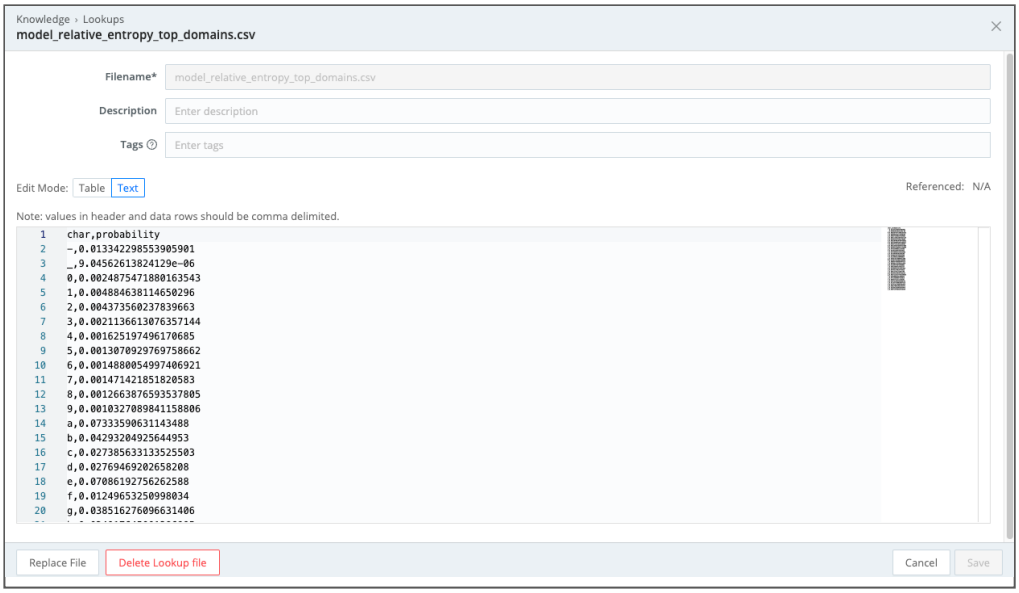
Modify Lookup Files

To re-upload, expand, edit, or delete an existing `.csv` or `.gz` lookup file, select its row on the Lookups page. (No editing option is available for `.mmdb` files; you can only re-upload or delete these.)

In the resulting modal, you can edit files in **Table** or **Text** mode. However, **Text** mode is disabled for files larger than 1 MB.



Editing in **Table** mode



Editing in **Text** mode

Edit Regex with Quotation Marks

When a regex contains quotes, edit it in **Text** mode. Do not edit it **Table** mode, which can cause the regex to behave in unexpected ways.

Memory Sizing for Large Lookups

For large lookup files, you'll need to provide extra memory beyond basic requirements for Cribl Edge and the OS. To determine how much extra memory to add to a Worker Node/Edge Node for a lookup, use this formula:

```
Lookup file's uncompressed size (MB) * 2.25 (to 2.75) *  numWorkerProcesses = Extra RAM required for lookup
```

For example, if you have a lookup file that is 1 GB (1,000 MB) on disk, and three Worker Processes, you could use an average 2.50 as the multiplier:

```
1,000 * 2.50 * 3 = 7,500
```

In this case, the Node's server or VM would need an extra 7,500 MB (7.5 GB) to accommodate the lookup file across all three worker processes.

What's with That Multiplier?

We've cited a squishy range of 2.25–2.75 for the multiplier, because we've found that it varies inversely with the number of columns in the lookup file:

- The fewer columns, the higher the extra memory overhead (2.75 multiplier).
- The more columns, the lower the overhead (2.25 multiplier).

In Cribl's testing:

- 5 columns required a multiplier of 2.75
- 10 columns required a multiplier of only 2.25.

These are general (not exact) guidelines, and this multiplier depends only on the lookup table's number of columns. The memory overhead imposed by each additional row appears to decline when more columns are present in the data.

Maximum Table Size

Aside from the memory requirements above, the Node.js runtime imposes a hard limit on the size of lookup tables that Cribl Edge can handle. No table can contain more than 16,777,216 (i.e., 2^{24}) rows. If a lookup exceeds this size, attempting to load the lookup will log errors of the form: `failed to load function...Value undefined out of range...`.

Other Scenarios

See also:

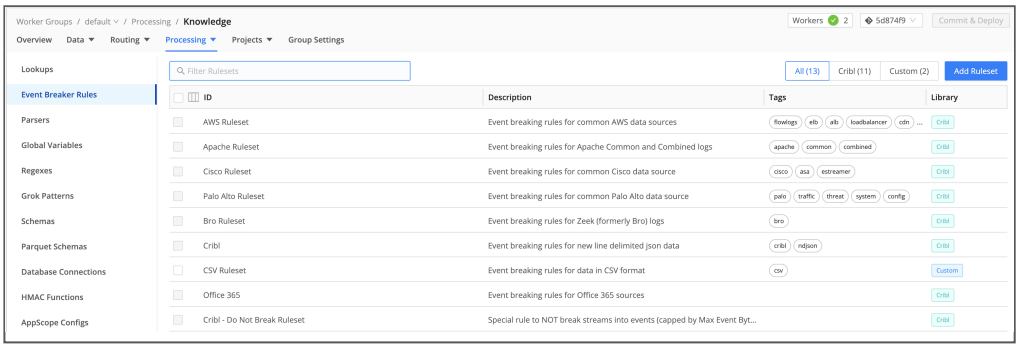
- Lookup Function.
- Ingest-time Lookups use case.
- Managing Large Lookups use case.
- Redis Function for faster lookups using a Redis integration (bypasses the Lookups Library).

14.2. Event Breakers

Event Breakers help break incoming streams of data into discrete events. To see how Event Breakers interact with the rest of Cribl Edge's data flow, see Event Processing Order.

Access Event Breakers

To access the Event Breakers management interface navigate to **Products** > **Cribl Edge** > **Fleets**. Select a Fleet, then go to **Processing** > **Knowledge** > **Event Breaker Rules**. On the resulting **Event Breaker Rules** page, you can edit, add, delete, search, and tag Event Breaker rules and rulesets, as necessary.



Event Breaker Rulesets page

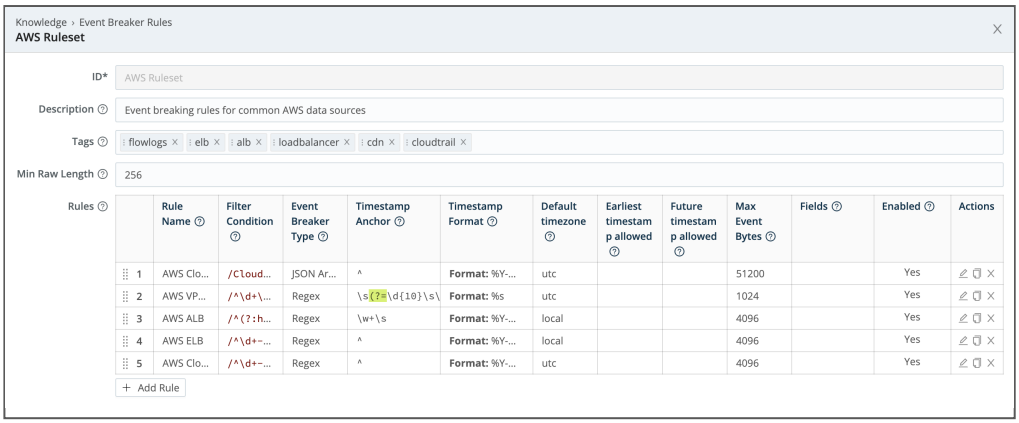
Limitations

Event Breakers are directly accessible only on Sources that require incoming events to be broken into a better-defined format. (Check individual Cribl Edge Sources' documentation for Event Breaker support.) Also, Event Breakers are currently not supported in Packs.

However, you can instead use the Event Breaker Function in Pipelines that process data from unsupported Sources, and in Pipelines within Packs.

Event Breaker Rulesets

Rulesets are **collections of Event Breaker rules** that are associated with Sources. Rules define configurations needed to break down a stream of data into events.



Rules within an example (AWS) ruleset that ships with Cribl Edge

Rules within a ruleset are ordered and evaluated top->down. One or more rulesets can be associated with a Source, and these rulesets are also evaluated top->down. For a stream from a given Source, the first matching rule goes into effect.

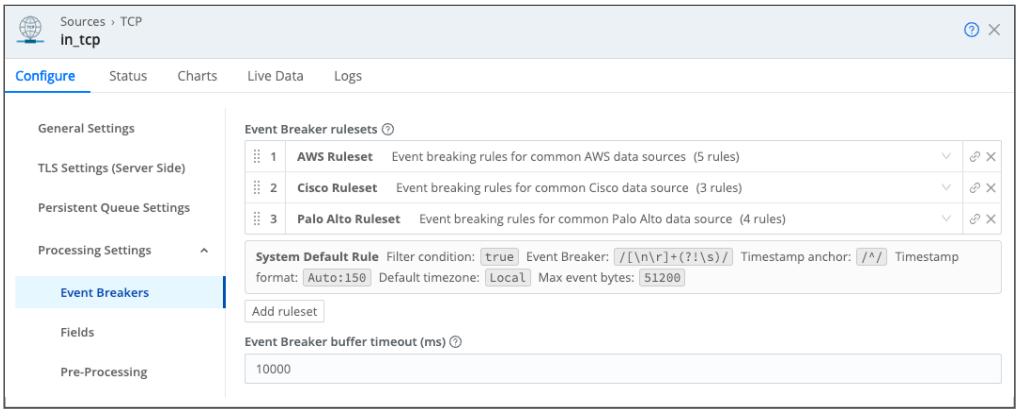
Rulesets and Rules - Ordered

```
Ruleset A
  Rule 1
  Rule 2
  ...
  Rule n

...

Ruleset B
  Rule Foo
  Rule Bar
  ...
  Rule FooBar
```

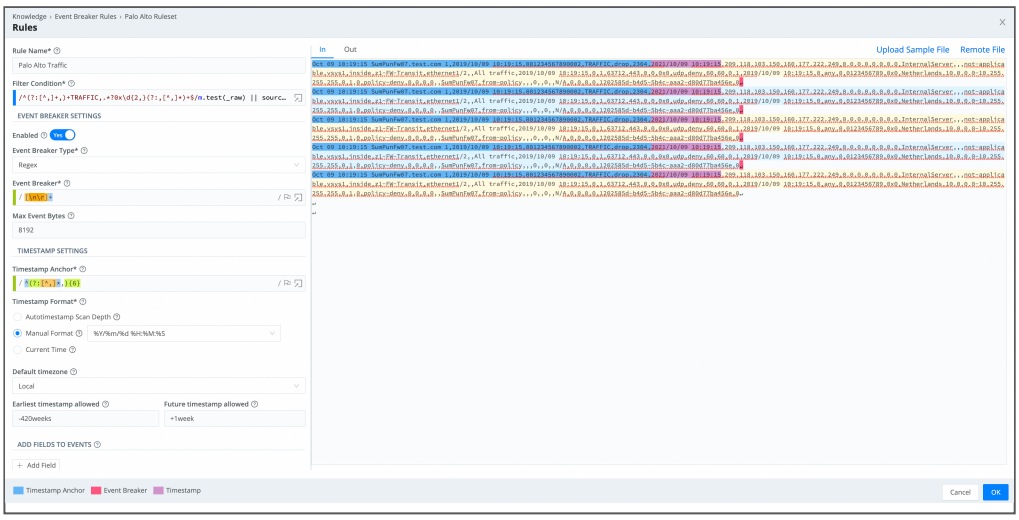
An example of multiple rulesets associated with a Source:



Three Event Breaker rulesets added to a Source

Rule Example

This rule breaks on newlines and uses Manual timestamping **after** the sixth comma, as indicated by this pattern: `^(?:[^,]*,){6}`.



An Event Breaker rule

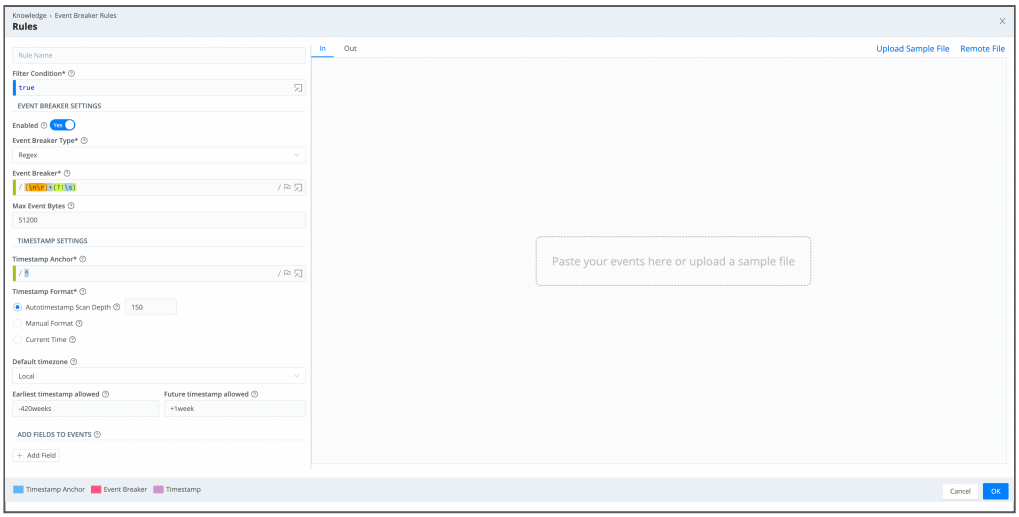
System Default Rule

The system default rule functionally sits at the bottom of the ruleset/rule hierarchy (but is built-in and not displayed on the Event Breakers page), and goes into effect if there are no matching rules:

- Filter Condition defaults to `true`
- Event Breaker to `[\n\r]+(?!\s)`
- Timestamp anchor to `^`
- Timestamp format to `Auto` and a scan depth of `150` bytes
- Max Event Bytes to `51200`
- Default Timezone to `Local`

How Do Event Breakers Work

On the **Event Breaker Rules** page (see screenshot above), select **Add Ruleset** to create a new Event Breaker ruleset. Select **Add Rule** within a ruleset to add a new Event Breaker.



Adding a new Event Breaker rule

Each Event Breaker includes the following components, which you configure from top to bottom in the above **Event Breaker Rules** modal:

Filter Condition

As a stream of data moves into the engine, a rule's filter expression is applied. If the expression evaluates to `true`, the rule configurations are engaged for the entire duration of that stream. Otherwise, the next rule down the line is evaluated.

The "stickiness" of the first `true` expression can lead to some unintended behavior. For example, if an S3 Source ingests an archive file that contains multiple files in different formats – where each file requires its own unique Event Breaker – events will not be broken correctly.

Event Breaker Type

After a breaker pattern has been selected, it will apply on the stream **continuously**. See below for specific information on different Event Breaker Types.

Timestamp Settings

After events are synthesized out of streams, Cribl Edge will attempt timestamping. First, a timestamp anchor will be located inside the event. Next, starting there, the engine will try to do one of the following:

- Scan up to a configurable depth into the event and autotimestamp, **or**
- Timestamp using a manually supplied `strptime` format, **or**
- Timestamp the event with the current time.

The closer an anchor is to the timestamp pattern, the better the performance and accuracy – especially if multiple timestamps exist within an event. For the manually supplied option, the anchor must lead the engine

right before the timestamp pattern begins.

Anchors preceding timestamps

This timestamping executes the same basic algorithm as the Auto Timestamp Function and the C.Time.timestampFinder() native method.

Cribl Edge truncates timestamps to three-digit (milliseconds) resolution, omitting trailing zeros.

In Cribl Edge 3.4.2 and above, where 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.

Add Fields to Events

After events have been timestamped, one or more fields can be added here 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 below omit the `cribl_breaker` field for brevity, but in real life the field is always added.)

Max Event Bytes

If an event's final broken chunk reaches a matched rule's **Max event bytes** length, it will be broken again.

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`.

Event Breaker Types

Several types of Event Breaker can be applied to incoming data streams:

- Regex

- File Header
- JSON Array
- JSON New Line Delimited
- Timestamp
- CSV

Regex

The Regex breaker uses regular expressions to find breaking points in data streams.

After a breaker regex pattern has been selected, it will apply on the stream **continuously**. Breaking will occur at the beginning of the match, and the matched content will be consumed/thrown away. If necessary, you can use a positive lookahead regex to keep the content – e.g.: `(?=pattern)`

Capturing groups are **not allowed** to be used anywhere in the Event Breaker pattern, as they will further break the stream – which is often undesirable. Breaking will also occur if the final broken event's length reaches the rule's **Max event bytes**.

Example

Break after a newline or carriage return, but only if followed by a timestamp pattern:
Event Breaker: `[\n\r]+(?=\d+-\d+-\d+\s\d+:\d+:\d+)`

Sample Event - Multiline

```
--- input ---
2020-05-19 16:32:12 moen3628 ipsum[5213]: Use the mobile TCP feed, then you can prog
   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 prog
   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!


--- output event 1 ---
{
  "_raw": "2020-05-19 16:32:12 moen3628 ipsum[5213]: Use the mobile TCP feed, then y
  "_time": 1589920332
}

--- output event 2 ---
{
  "_raw": "2020-05-19 16:32:12 moen3628 ipsum[5213]: Use the mobile TCP feed, then y
  "_time": 1589920332
}
```

File Header

You can use the File Header breaker to break files with headers, such as IIS or Bro logs. This type of breaker relies on a header section that lists field names. The header section is typically present at the top of the file, and can be single-line or greater.

After the file has been broken into events, fields will also be extracted, as follows:

- **Header Line**: Regex matching a file header line. For example, `^#`.
- **Field Delimiter**: Field delimiter regex. For example, `\s+`.
- **Field Regex**: Regex with one capturing group, capturing all the fields to be broken by field delimiter. For example, `^#[Ff]ields[:]?\s+(.*)`
- **Null Values**: Representation of a null value. Null fields are not added to events.
- **Clean Fields**: Whether to clean up field names by replacing non `[a-zA-Z0-9]` characters with `_`.

Example

Using the values above, let's see how this sample file breaks up:

Sample Event - File Header

```
--- input ---
#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


--- output event 1 ---
{
  "_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
}

--- output event 2 ---
{
  "_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",
  "_time": 1331904609.19
}
```

JSON Array

You can use the JSON Array to extract events from an array in a JSON document (e.g., an Amazon CloudTrail file).

- **Array Field**: Optional path to array in a JSON event with records to extract. For example, `Records`.
- **Timestamp Field**: Optional path to timestamp field in extracted events. For example, `eventTime` or `level1.level2.eventTime`.

- **JSON Extract Fields**: Enable this toggle to auto-extract fields from JSON events. If disabled, only `_raw` and `time` will be defined on extracted events.
- **Timestamp Format**: If **JSON Extract Fields** is set to `No`, you **must** set this to **Autotimestamp** or **Current Time**. If **JSON Extract Fields** is set to `Yes`, you can select any option here.

Example

Using the values above, let's see how this sample file breaks up:

Sample Event - JSON Document (Array)

```
--- input ---
{"Records":[{"eventVersion":"1.05","eventTime":"2020-04-08T01:35:55Z","eventSource":"
{"eventVersion":"1.05","eventTime":"2020-04-08T01:35:56Z","eventSource":"ec2.amazonaw

--- output event 1 ---
{
  "_raw": "{\"eventVersion\":\"1.05\",\"eventTime\":\"2020-04-08T01:35:55Z\",\"events
  "_time": 1586309755,
  "cribl_breaker": "j-array"
}

--- output event 2 ---
{
  "_raw": "{\"eventVersion\":\"1.05\",\"eventTime\":\"2020-04-08T01:35:56Z\",\"events
  "_time": 1586309756,
  "cribl_breaker": "j-array"
}
```

JSON New Line Delimited

You can use the JSON New Line Delimited breaker to break and extract fields in newline-delimited JSON streams.

Example

Using default values, let's see how this sample stream breaks up:

Sample Event - Newline Delimited JSON Breaker

```
--- input ---
{"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"


--- output event 1 ---
{
  "_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,
}

--- output event 2 ---
{
  "_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,
}
```

Timestamp

You can use the Timestamp breaker to break events at the beginning of any line in which Cribl Edge finds a timestamp. This type enables breaking on lines whose timestamp pattern is not known ahead of time.

Example

Using default values, let's see how this sample stream breaks up:

Sample Event - Timestamp Based Breaker

```
--- input ---
{"level":"debug","ts":"2021-02-02T10:38:46.365Z","caller":"sdk/sync.go:42","msg":"Ha
{"level":"debug","ts":"2021-02-02T10:38:56.365Z","caller":"sdk/sync.go:42","msg":"Ha


--- output event 1 ---
{
  "_raw": "{\"level\":\"debug\",\"ts\":\"2021-02-02T10:38:46.365Z\",\"caller\":\"sdk/
  "_time": 1612262326.365
}

--- output event 2 ---
{
  "_raw": "{\"level\":\"debug\",\"ts\":\"2021-02-02T10:38:56.365Z\",\"caller\":\"sdk/
  "_time": 1612262336.365
}
```

CSV

The CSV breaker extracts fields in CSV streams that include a header line. Selecting this type exposes these extra fields:

- **Delimiter**: Delimiter character to use to split values. Defaults to: `,`
- **Quote Char**: Character used to quote literal values. Defaults to: `"`
- **Escape Char**: Character used to escape the quote character in field values. Defaults to: `"`
  Example: Using default values, let's see how this sample stream breaks up:

Example

Using default values, let's see how this sample stream breaks up:

Sample Event - CSV Breaker

```
--- input ---
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


--- output event 1 ---
{
  "_raw": "\"1611768713\",\"myHost1\",\"anet\",\"cisco\",\"ASN4204269\",\"11430\",\"
  "time": "1611768713",
  "host": "myHost1",
  "source": "anet",
  "model": "cisco",
  "serial": "ASN4204269",
  "bytes_in": "11430",
  "bytes_out": "43322",
  "cpu": "0.78",
  "_time": 1611768713
}

--- output event 2 ---
{
  "_raw": "\"1611768714\",\"myHost2\",\"anet\",\"cisco\",\"ASN420423\",\"345062\",\"
  "time": "1611768714",
  "host": "myHost2",
  "source": "anet",
  "model": "cisco",
  "serial": "ASN420423",
  "bytes_in": "345062",
  "bytes_out": "143433",
  "cpu": "0.28",
  "_time": 1611768714
}
```

With **Type: CSV** selected, an Event Breaker will properly add quotes around all values, regardless of their initial state.

Cribl Versus Custom Rulesets

Event Breaker rulesets shipped by Cribl will be listed under the **Cribl** tag, while user-built rulesets will be listed under **Custom**. Over time, Cribl will ship more patterns, so this distinction allows for both sets to grow independently. In the case of an ID/Name conflict, the Custom pattern takes priority in listings and search.

Export and Import Rulesets

You can export and import Custom (or Cribl) rulesets as JSON files. This facilitates sharing rulesets among Worker Groups 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. You can now modify the ruleset directly in JSON, if you choose.
4. 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.

Edge: Build Breakers from Files

You can use Cribl Edge to apply and author Event Breakers while exploring files – even 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.

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.

Specify Minimum `_raw` Length

If you notice that the same Collector or Source is applying inconsistent Event Breakers to different datasets – e.g., 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.

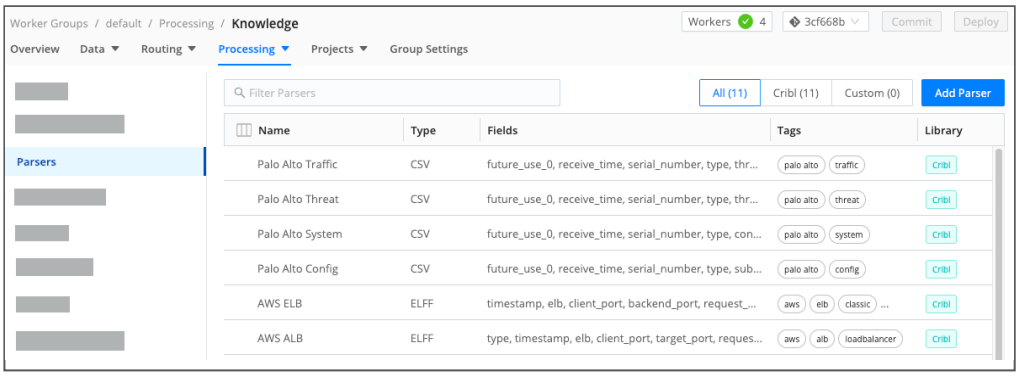
You can tune this **Min Raw Length** threshold to account for varying numbers of inapplicable characters at the beginning of events. The default is `256` characters, but you can set this as low as `50`, or as high as `100000` (100 KB).

14.3. Parsers Library

What Are Parsers

Parsers are definitions and configurations for the Parser Function. The purpose of the Parsers Library is to provide an interface for creating and editing Parsers. The library is searchable, and each parser can be tagged as necessary.

To find the library, select **Fleets** from the sidebar, and choose a Fleet. Then, on the **Fleets** submenu, select **More**, then **Knowledge**, then **Parsers**.



Parsers Library

Parsers can be used to **extract** or **reserialize** events. See Parser Function page for examples.

Supported Parser Types:

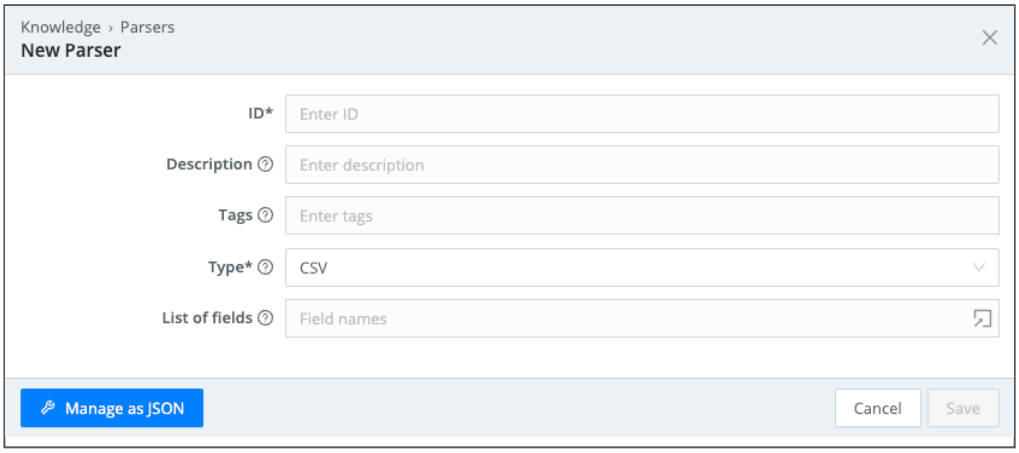
- CSV – Parse and reserialize comma-separated values.
- ELFF – Parse and reserialize events in Extended Log File Format.
- CLF – Parse and reserialize events in Common Log Format.

Creating a Parser

To create a parser, follow these steps:

1. In the sidebar, select **Worker Groups** (Cribl Stream) or **Fleets** (Cribl Edge) from the sidebar, and choose a Worker Group or Fleet.
2. On the **Worker Groups/Fleets** submenu, select **Processing** (Cribl Stream) or **More** (Cribl Edge), 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.
8. Optionally, enter any desired **Tags**.



Adding a new parser

14.4. Global Variables Library

What Are Global Variables

Global Variables are reusable JavaScript expressions that can be accessed in Functions in any Pipeline.

To access the library, select **Fleets** from the sidebar, and choose a Fleet. Then, on the **Fleets** submenu, select **More**, then **Knowledge**, then **Global Variables**.

Typical use cases for Global 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.

Global Variables can be of the following types:

- Number
- String
- Boolean
- Object
- Array
- Expression

Global Variables can be accessed via `C.vars.` – which can be called anywhere in Cribl Edge that JS expressions are supported. Typeahead is provided. More on Cribl Expressions here.

Examples

Scenario 1

Assign field `foo` the value in `theAnswer` Global Variable.

- Global Variable Name: `theAnswer`
- Global Variable Value: `42`
- **Sample Eval Function:** `foo = C.vars.theAnswer`

Scenario 2

Assign field `nowEpoch` the current time, in epoch format.

- Global Variable Name: `epoch`
- Global Variable Value: `Date.now()/1000`
- **Sample Eval Function:** `nowEpoch = C.vars.epoch()`

Scenario 3

Create a new field called `storage`, by converting the value of event field `size` to human-readable format.

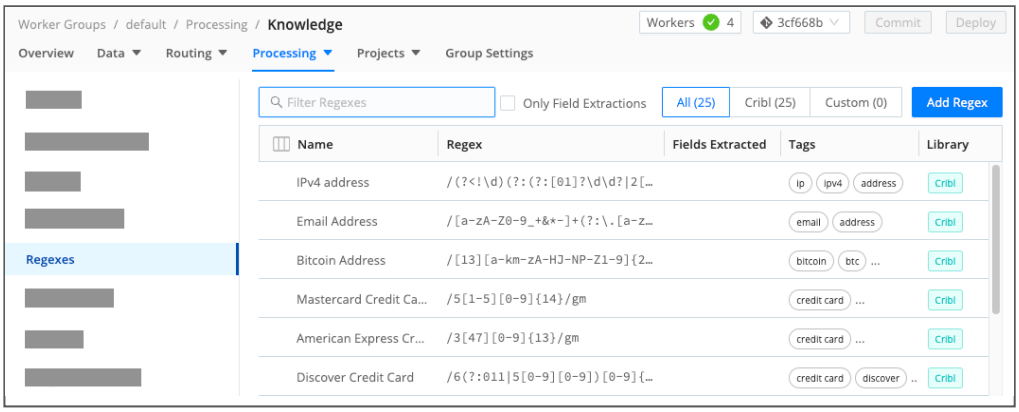
- Global Variable Name: `convertBytes`
- Global Variable Value: `` `${Math.round(bytes / Math.pow(1024, (Math.floor(Math.log(bytes) / Math.log(1024)))), 2)}${['Bytes', 'KiB', 'MiB', 'GiB', 'TiB', 'PiB', 'EiB', 'ZiB', 'YiB'][(Math.floor(Math.log(bytes) / Math.log(1024)))]}` ``
  Note the use of quotes or backticks around values. Use the opposite delimiter for the enclosing expression.
- Global Variable Argument: `bytes`
- **Sample Eval Function:** `storage = C.vars.convertBytes(size)`
  Note the use of `bytes` here as an argument.

14.5. Regex Library

What Is the Regex Library

Cribl Edge ships with a Regex Library that contains a set of pre-built common regular expression patterns. This library serves as an easily accessible repository of regular expressions. The Library is searchable, and you can assign tags to each pattern for further organization or categorization.

To access the library, select **Fleets** from the sidebar, and choose a Fleet. Then, on the **Fleets** submenu, select **More**, then **Knowledge**, then **Regexes**.

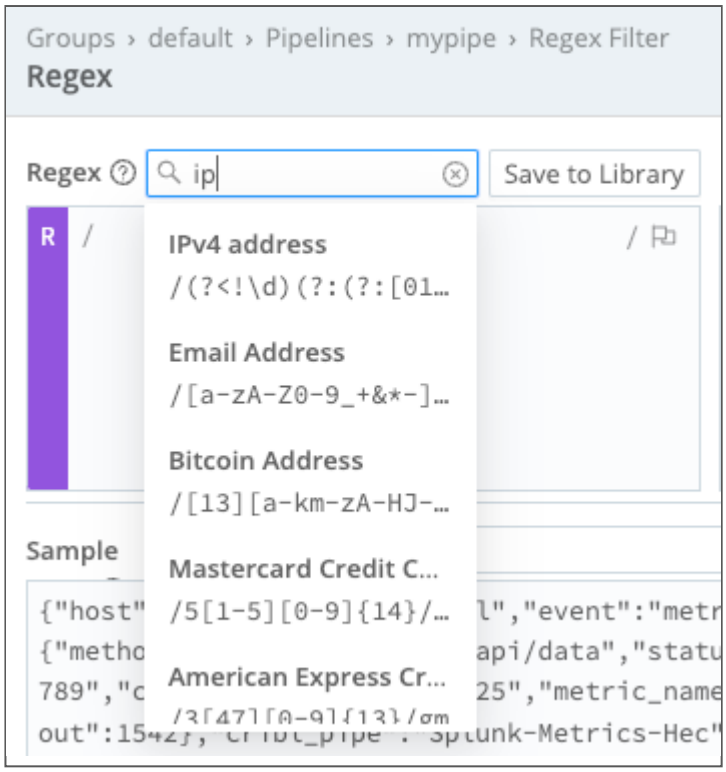


Regex Library

Using Library Patterns

To insert a pattern into a Function's regex field, first select the Advanced mode or Edit icon beside that field.

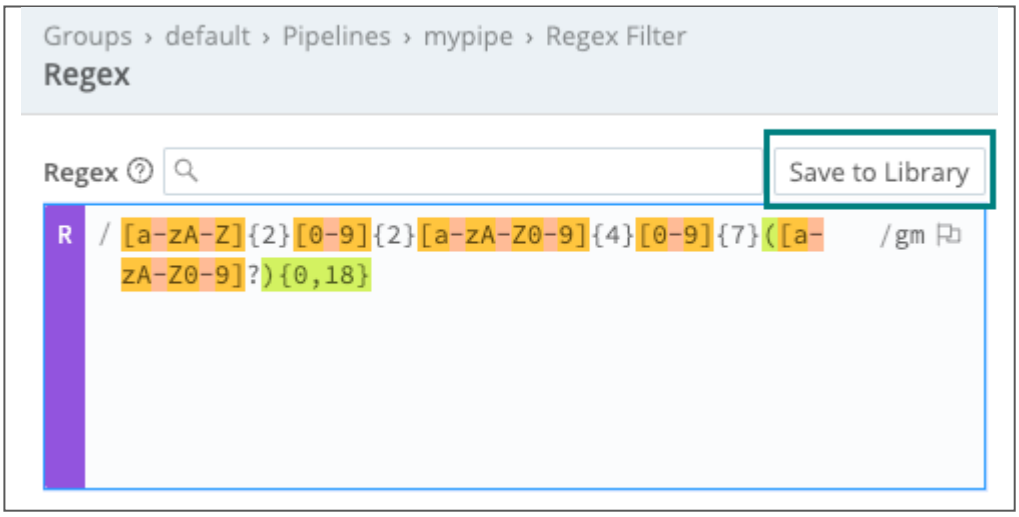
In the resulting Regex or Rules modal, Regex Library patterns will appear as typeahead options. Select a pattern to paste it in. You can then use the pattern as-is, or modify it as necessary.



Inserting a pattern from the Regex Library

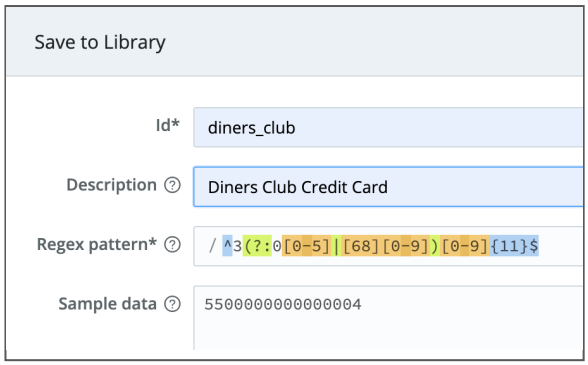
Adding Patterns to the Library

You can also add new, custom patterns to the Library. In the same modal, once you've built your pattern, select **Save to Library**.



Adding a custom pattern to the Regex Library from a Function's Regex modal

In the resulting modal, give your custom pattern a unique ID. Optionally, you can also provide a **Description** (name) and groom the **Sample data**. Then select **Save**.



Identifying the custom pattern

Your custom pattern will now reside in the Regex Library. It will be available to Functions using the same typeahead assist as Cribl's pre-built patterns.

Cribl vs. Custom and Priority

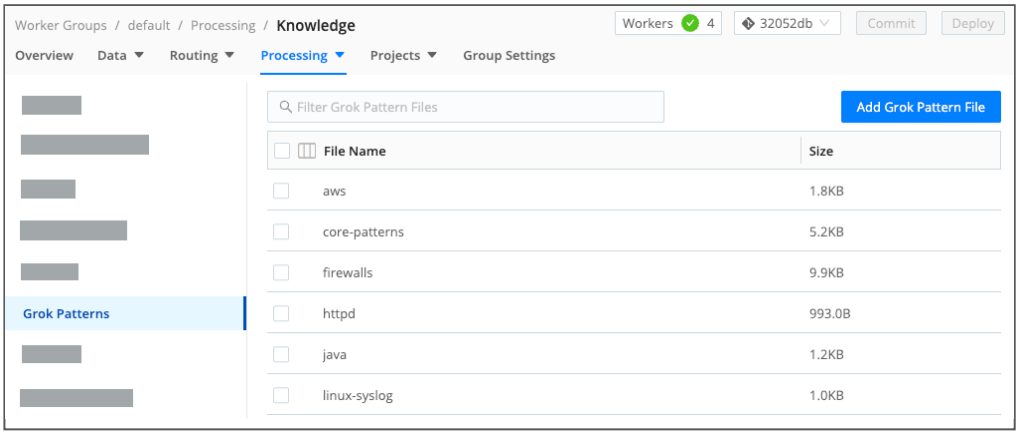
Within the Library, patterns shipped by Cribl will be listed under the **Cribl** tab, while those built by users will be found under **Custom**. Over time, Cribl Edge will ship more patterns, and this distinction allows for both sets to grow independently.

In the case of an ID/Name conflict, 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.

14.6. Grok Patterns Library

What Is the Grok Patterns Library

Cribl Edge ships with a Grok Patterns Library that contains a set of pre-built common patterns, organized as files.



Grok Patterns Library

Managing Library Patterns

You can access the Grok Patterns Library by selecting **Fleets** from the sidebar and choosing a Fleet. Then, on the **Fleets** submenu, select **More**, then **Knowledge**, then **Grok Patterns**. The library contains several pattern files that Cribl provides for basic Grok scenarios, and is searchable.

To create a new pattern file, select **Add Grok Pattern File**. In the resulting modal, assign a unique **File name**, populate the file with patterns, then select **Save**.

Pattern files reside in: `$CRIBL_HOME/(default|local)/cribl/grok-patterns/`

Using Grok Patterns

In the current Cribl Edge version, you apply Grok patterns by inserting a Grok Function into a Pipeline, then manually typing or pasting patterns into the **Pattern** field(s).

14.7. Schemas Library

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 obviously serve different purposes. Beware of assuming that operations that work for one kind of schema can be used with the other. For example, the validation method for JSON schemas – `C.Schema('<schema_name>').validate(<object_field>)` – can't be used to validate Parquet schemas.

JSON Schemas

JSON schemas are based on the popular JSON Schema standard, and Cribl Edge supports schemas matching that standard's Drafts 0 through 7.

To access the schema library, select **Fleets** from the sidebar, and choose a Fleet. Then, on the **Fleets** submenu, select **More**, then **Knowledge**, then **Schemas**.

Its purpose is to provide an interface for creating, editing, and maintaining schemas.

You validate a schema using this built-in method:
`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:

- Making a decision before sending an event down to a destination.
- Making a decision before accepting an event (for example, drop an event if invalid).
- Making a decision to route an event based on the result of validation.

JSON Schema Example

To add this example JSON Schema, on the **Schemas** page, select **Add Schema**. Enter the following:

- 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
    }
  }
}
```

Assume that events look like this:

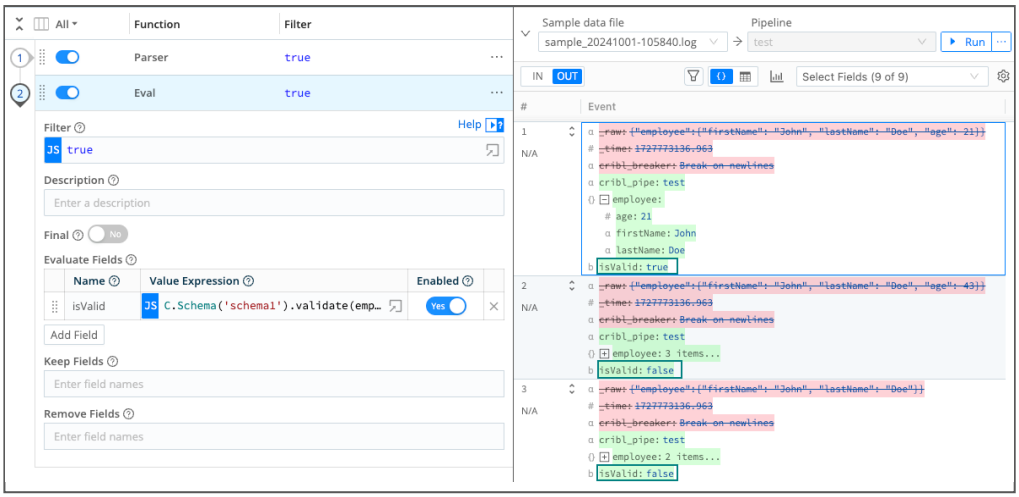
Events

```
{"employee":{"firstName": "John", "lastName": "Doe", "age": 21}}
{"employee":{"firstName": "John", "lastName": "Doe", "age": 43}}
{"employee":{"firstName": "John", "lastName": "Doe"}}
```

To validate whether the `employee` field is valid per `schema1`, we can use the following:
`C.Schema('schema1').validate(employee)`

Results:

- First event is **valid**.
- Second event is **not valid** because `age` is greater than the maximum of `42`.
- Third event is **not valid** because `age` is missing.



Schema validation results for the above events

14.8. Parquet Schemas

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 obviously serve different purposes. Beware of assuming that operations that work for one kind of schema can be used with the other. For example, the validation method for JSON schemas – `C.Schema('<schema_name>').validate(<object_field>)` – can't be used to validate Parquet schemas.

Configuring Parquet Schemas

Destinations whose **General Settings** > **Data format** drop-down includes a `Parquet` option can write out data as files in the Apache Parquet columnar storage format.

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.

If you turn on a Destination's **Parquet Settings** > **Automatic schema**, Cribl Edge will automatically generate a Parquet schema based on the events of each Parquet file it writes. The automatically-generated schema preserves the exact structure of the ingested events, with no data lost. In Cribl Edge v.4.4.2, every field in the generated schema will have a data type of `JSON`, `STRING`, `INT_64`, `UNSIGNED_INT_64`, or `DOUBLE`.

(The one exception is the Amazon Security Lake Destination, where automatic schema generation is not relevant because Amazon Security Lake exclusively uses the OCSF Parquet schemas already available in the drop-down.)

Pros and Cons of Automatically Generated Parquet schemas

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.

Configuring Parquet Schemas

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.

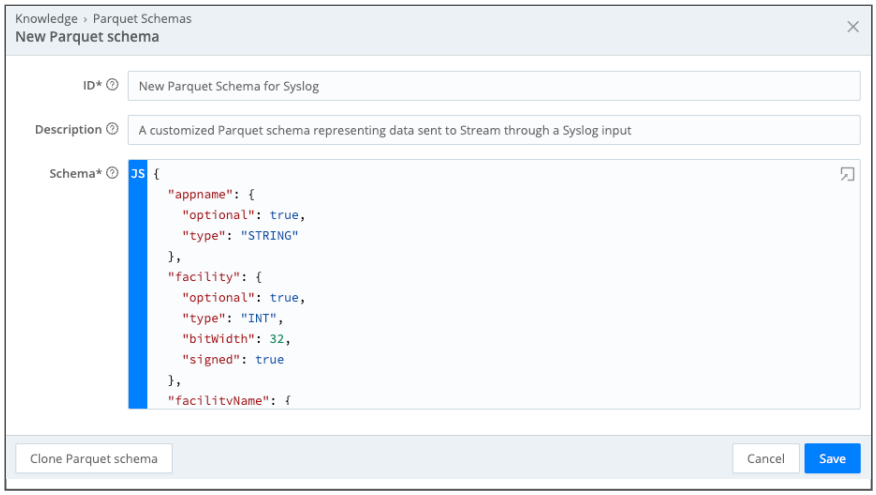
You do not need to start from scratch: Cribl provides sample Parquet schemas for you to clone and then customize as needed.

To access the Parquet schema library, select **Fleets** from the sidebar, and choose a Fleet. Then, on the **Fleets** submenu, select **More**, then **Knowledge**, then **Parquet Schemas**.

To add a Parquet schema, select **Add Parquet schema** to open the **New Parquet schema** modal.

You can also clone an existing Parquet schema to create a new schema:

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

Now, when you configure your Destination, the schema you created will be available from the **Parquet Settings** > **Parquet schema** drop-down.

Creating/Modifying Parquet Schemas from Destinations

When the existing schemas on a Destination's **Parquet schema** drop-down don't meet your needs, follow this procedure to add or modify (clone) a new schema.

1. Navigate to the Parquet Schemaa Library at **Processing** > **Knowledge** > **Parquet Schemas**.
2. Add or modify the schema, as described above.

To modify an existing schema, Cribl strongly recommends that you first clone the schema, and give the clone its own distinct name. See Cloning Is Safer just below.

3. Return to the Destination, and select the new or modified schema from the **Parquet schema** drop-down.

Cloning Is Safer than Modifying

You can modify an existing schema, but Cribl strongly recommends that you first clone the schema, and give the clone its own distinct name. When you clone a schema, return to the Destination, and select the clone from the **Parquet schema** drop-down.

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 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.

Working with Parquet in Cribl Edge

Different Parquet readers and writers behave differently. Keep the following guidelines in mind when working with Parquet in Cribl Edge.

The Parquet Schema Editor

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

For now, 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 Schema Expressed 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);
    }
  }
}
```

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"
          }
        }
      }
    }
  }
}
```

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);
    }
  }
}
```

Parquet Schema Example 2 Expressed as JSON

```
{
  "the_map": {
    "type": "MAP",
    "fields": {
      "key_value": {
        "repeated": true,
        "fields": {
          "key": {
            "type": "STRING"
          },
          "value": {
            "type": "STRING"
          }
        }
      }
    }
  }
}
```

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;
        }
      }
    }
  }
}
```

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",
            "fields": {
              "list": {
                "repeated": true,
                "fields": {
                  "element": {
                    "type": "INT64"
                  }
                }
              }
            }
          }
        }
      }
    }
  }
}
```

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.

14.9. AppScope Configs

You can obtain high-fidelity application data from any Linux command or application using AppScope, an open-source tool from Cribl. This includes data that's difficult or impossible to obtain any other way. (Note that AppScope is no longer being actively developed by Cribl.)

AppScope is included in your Cribl Edge installation. One or two AppScope Sources are enabled right out-of-the-box; they are configured to send data to Cribl Edge or Cribl Stream on the same host.

AppScope ships with a set of config files that offer a variety of options for what events and metrics you want AppScope to capture, and whether to enable payload capture. These are: `sample_config`, `default_metrics_events`, `verbose_metrics_events`, and `verbose_metrics_events_payloads`.

To understand precisely what one of these files does, select it to open its **AppScope Configs** modal and select **Manage as JSON**. For explanations and definitions, see the Config File and Schema Reference sections in the AppScope docs.

You can customize the config files, or edit them via **Manage as JSON**. When editing, you can revert to the original state by selecting **Restore Default**. The safer option is to select **Clone AppScope config**, then edit and save your new version with a new file name. The sample configs, and any variants you have created and saved, constitute a reusable set of AppScope configurations in the AppScope Config Library.

When you have assigned an AppScope config to a Rule in an AppScope Source, and you change it in the Knowledge UI, the changes will not take effect until you do the following:

- In the AppScope Source **AppScope Rules** settings, re-assign the config to the Rule. (That is, unselect the config file, then select it again.)
- Save and commit.

Creating AppScope Configs

To access Appscope Configs:

1. In the sidebar, select **Fleets**, and choose a Fleet.
2. Then, on the **Fleets** submenu, select **More**, then **Knowledge**, then **Appscope Configs**.

Select **Add AppScope config** to open a modal to begin creating an AppScope config, using the options below.

Once you've created AppScope configs, you can edit, delete, search, and tag them as desired.

General Settings

ID: Enter a unique name to identify this AppScope Config.

Description: Optionally, enter a description that will remind you what the config is for when you see it listed in the library.

Tags: Optionally, enter any desired tags.

Transports

AppScope offers several options for routing data. See Data Routing in the AppScope docs.

Use AppScope Source's IP/port or UNIX domain socket?: The default `Yes` setting applies the transport options defined in the associated AppScope Source. Toggling this to `No` exposes the following fields in this section.

Cribl authentication token: Optionally, set a value to add to the `config-event` header (as a top-level `authToken` property) when AppScope establishes a connection.

Send metrics and events over the same connection?: Defaults to `Yes`, which locks some settings into preconfigured values. Toggle to `No` if you prefer to route metrics and events independently.

Connection type: From the drop-down, choose one of the connection types described below, then enter values in whatever fields the UI exposes.

- When **Send metrics and events over the same connection?** is toggled to `Yes`, you configure a single connection by choosing `TCP`, `Edge` (the default), or `Unix`.
- When routing metrics and events independently, you configure one connection for metrics and another for events; the drop-down offers two additional connection types: `UDP` and `File`.

Transports and Connection Types

At the heart of any AppScope config are the transports you define. In AppScope, a transport specifies something to send, how to send it, and where to send it. The things to send can be metrics, events, or logs.

Depending on the choices you make in the **Transport** settings, the UI will make some or all of the following types of connections available.

UDP: The destination is the network server whose hostname or IP address you specify in the **Host** field, along with a UDP **Port** to listen on.

TCP: The destination is the network server whose hostname or IP address you specify in the **Host** field, along with a TCP **Port** to listen on. The **Enable TLS** toggle defaults to `No`.

Unix: The destination is a Unix domain server listening on the Unix domain **Socket path** you specify.

File: The destination is the file whose directory location you specify in the **File path** field, along with a **Buffering** method.

Edge: The destination is Cribl Edge, listening on a preconfigured Unix domain socket.

TLS Settings (TCP Only)

When the **Connection type** is `TCP`, the UI exposes the **Enabled** toggle, which defaults to `No`.

If you toggle **Enabled** to `Yes`:

- Set **Validate server certs** to `true` if you want the connection to fail when the TLS server certificate cannot be validated. Defaults to `No`.
- Leave **CA certificate path** blank if you toggled the **Validate server certs** to `true` and are using the local OS-provided trusted CA certificates to validate the server's certificate. Otherwise, specify the server path containing CA certificates, which must be in PEM format.

Logs Settings

Configure where and how AppScope should send its own logs.

Connection type: From the drop-down, choose one of the connection types described above, then enter values in whatever fields the UI exposes.

Log level: Set logging verbosity. When **Send metrics and events over the same connection?** is toggled to `Yes`, **Log level** is locked at `warning`.

File path: Specify a directory in which to store logs. If not specified, defaults to `/tmp/scope.log`.

Buffering: Set this method to `Line` if there's a chance that multiple scoped processes will be writing to the same file. This prevents the log file from getting scrambled, with interleaved lines. However, in single-writer scenarios, selecting `Full` often improves performance.

Metrics Settings

The Schema Reference in the AppScope docs lists and fully describes every metric that AppScope can emit.

Process (`proc.*`) metrics periodically report information about resource usage at a point in time. All other metrics report on actions taken by a scoped process. For examples, see Metrics and Events in AppScope in the AppScope documentation.

Should AppScope emit metrics?: Defaults to `Yes`, which exposes the following options.

Format: AppScope can emit metrics in either of two formats: `NDJSON` (the default), or `StatsD`. If you choose `StatsD`, the UI exposes the following fields:

- **StatsD prefix**: Optionally, enter a prefix for AppScope to add to metrics.
- **StatsD max length**: Optionally, enter a maximum length, in bytes, for StatsD-formatted metrics. AppScope will truncate any metric which exceeds the maximum length. (Defaults to `512`.)

Verbosity: Set a value between `0` and `9`. (Defaults to `4`.) Verbosity controls both the cardinality of metrics that AppScope emits, and which kinds of metrics AppScope aggregates. For details, see Metrics and Events in AppScope in the AppScope documentation. But the general principle is:

- At low verbosity, metrics summarize multiple actions over a reporting period.
- At higher verbosity, metrics provide details about single actions in real time.

Reporting period: Set the metric summary interval in seconds. Defaults to `10`.

Watch list: Select categories of metrics to emit. Defaults to the following categories: `StatsD`, `File System`, `Network`, `HTTP`, `DNS`, `Process`.

Events Settings

The Schema Reference in the AppScope docs lists and fully describes every event that AppScope can emit.

Should AppsScope emit events?: Defaults to `Yes`, which exposes the following controls in this section.

Event-rate limiter (events per second): AppScope discards events generated in excess of the limiter setting, which defaults to `10000`. To disable the limiter, set it to `0`.

Enhance filesystem event data: When toggled to `Yes` (the default), AppScope enhances `fs.open` events by adding the `uid`, `gid`, and `mode` fields.

Watch list: Specify categories of events to emit, and configure their content and structure. Defaults to the following categories: `Console`, `File`, `Filesystem`, `Network`, `DNS`, and `HTTP`. For details, see Categories.

Events Categories

The **Watch list** is a table providing a row for each enabled category of events. The **Type** column contains the category names. The other columns (**Name**, **Field**, and **Value**) contain regular expressions.

At the right edge of the **Name**, **Field**, and **Value** fields is a Copy button, a Flag button to append Regex flags, and an Expand button to open a validation modal.

AppScope emits an event when its values match all the regexes in the corresponding category's row. For **HTTP**, request and response events must also match the headers listed in the **Header** column.

Allow binary is a toggle, available only for **Console** events. When toggled to `Yes` (the default), **Console** events will include binary data along with text data. If you prefer to redact binary data, toggle to `No`.

The **Type** column enumerates any or all of the following event category names. Toggle **Enabled** to `Yes` to include a category.

- **Console**: Includes writes to standard out and error. Use this to monitor console output, especially in containerized environments where logging to files isn't commonly done.
- **File**: Includes writes to files. Use this to monitor log files and/or any other files of interest.
- **Filesystem**: Includes filesystem operations like `open`, `close`, and `delete`.
- **Network**: Includes `open` and `close` events on network connections.
- **DNS**: Includes DNS request and response events.
- **HTTP**: Includes HTTP request and response events.
- **Granular**: Seldom used, and not emitted by default. Can be useful when tracking down a problem. When this type is enabled, AppScope sends more events than it otherwise would, giving you a more granular view of the activity of the scoped process.

Some usage examples:

- For `fs.*` events, AppScope normally emits `fs.open` and `fs.close` events. With **Granular** enabled, every time a read or write occurs, AppScope emits `fs.read` or `fs.write`, respectively.
- For `net.*` events, AppScope normally emits `net.open`, `net.close`, and `net.app`.With **Granular** enabled, every time a read or write is done on the socket, AppScope emits `net.rx` or `net.tx`, respectively.

Payloads

AppScope never sends encrypted payloads to disk, to Cribl Stream, or to Cribl Edge.

Capture all payloads: Enable capturing all payloads. Defaults to `No`. Enable this setting with caution: when scoping I/O-intensive programs, it can produce large amounts of data.

Protocol detection: Select **Add Protocol** to specify a protocol (for example, StatsD, HTTP, Redis) for AppScope to detect in network payloads, as well as what AppScope should do upon detecting that protocol. AppScope checks the first packet on a channel against the regular expression for each protocol until it finds a match. Then AppScope treats the matching protocol as the detected protocol for its channel, and ignores any others listed. See Sample Protocols for regexes that detect common protocols.

Name: Enter a short, descriptive name to define the protocol-detect event. AppScope inserts this value into the header for payloads sent to Cribl. Since AppScope sends this as JSON, you cannot include double quotes or backslashes.

Regex: Enter a regular expression to detect the desired protocol.

Generate protocol-detect-events: Toggle to `Yes` to generate protocol-detect events. Defaults to `No`.

Capture payload: Toggle to `Yes` to send payloads from matching streams to the configured destinations. Defaults to `No`.

Convert to hex: When toggled to `Yes` (the default), AppScope converts part of the payload to a hex-string before applying the regex. Otherwise, AppScope applies the regex to the binary payload.

Bytes to convert: The number of bytes to convert to hex before applying the regex. Defaults to `256`.

Payload directory: Specify the directory where AppScope should store payload files. Consider using a performant filesystem to reduce I/O performance impacts. Defaults to `tmp`. When **General Settings** > **Send metrics and events over the same connection?** is toggled to `Yes`, AppScope ignores this field and sends payloads to the default destination.

Advanced Settings

Send process-start message?: When toggled to `Yes` (the default), enables the process-start message. This message is the first one AppScope sends upon establishing a connection. It describes AppScope's configuration for that session, as well as the application being scoped.

Command directory: If you want to dynamically change AppScope configuration while scoping a running process, save a `scope.<pid>` file to a **command directory** of your choice, where `<pid>` is the PID of the scoped process, and the file contains lines of the form `<ENV_VAR>=<value>`. Once per reporting period, AppScope looks for a `scope.<pid>` file in the **command directory**, which defaults to `/tmp`. You can configure reporting period in the **Metrics** tab.

Metadata: If you want to add a field to events and metrics that AppScope emits, select **Add Metadata** and define it as a key-value pair. Key/value definitions cannot include environment variables.

Per-process configs: Optionally, define a config that, given a process that fits a particular pattern, will partly or completely override the config you have defined already.

Select **Add config** and configure the **Filters** relevant to the process(es) you want to match:

- **Process name**: Enter a string to match the basename of the scoped process.
- **Process argument**: Enter an expression to match the scoped process's full command line including options and arguments.
- **Hostname**: Enter a string to match the hostname of the machine where the scoped process is running.
- **Username**: Enter a string to match the username for the scoped process's UID.
- **Environment**: To match when an environment variable is set, enter its name alone (for example, `FOO`). To match when the variable is set with a particular value, enter both (for example, `FOO=bar`).
- **Ancestor**: Enter a string to match the basename of the scoped process's parent, parent's parent, and so on.

Then, in the **Config** column, select **Edit**. This takes you through the settings, beginning with **General Settings**, but this time you'll be specifying what in the previously-defined settings to override when a process matches your **Filters**.

Protocol Detection Examples for Payloads

Here are some example definitions for detecting protocols in payloads.

Redis

This regex detects the plain-text Redis serialization protocol (RESP).

Name: `Redis`

Regex: `"^[*]\\d+|^[+]\\w+|^[$]\\d+"`

Mongo

This regex detects the MongoDB wire protocol. Since it's binary, we convert to hex.

Name: `Mongo`

Regex: `^240100000000000000000000d407`

Convert to hex: `Yes`

Convert length: `14`

HTTP

By default, AppScope uses an internally-defined protocol detector for HTTP similar to this example.

Name: `HTTP`

Regex: `"HTTP\\/1\\.[0-2]|PRI \\* HTTP\\/2\\.0\r\n\r\nSM\r\n\r\n"`

StatsD

By default, AppScope uses an internally-defined protocol detector for StatsD similar to this example.

Name: `STATSD`

Regex: `"^([^:]+):([\\d.]+)\\|(c|g|ms|s|h)"`

TLS

By default, AppScope uses an internally-defined protocol detector for TLS/SSL similar to this example.

Name: `TLS`

Regex: `"^(?:(?:16030[0-3].{4})|(?:8[0-9a-fA-F]{3}01))"`

Convert to hex: `Yes`

Convert length: `5`

15. Reference

15.1. API Docs

To complement our API Reference, below are some examples of using the Cribl Edge API to address common scenarios. If you want to automate an action in Cribl Edge for which you can't find documentation, ask us in Cribl Community's `#docs` channel.

Breaking Change – v.4.0+

In Cribl Edge 4.0 and later, `PATCH` requests to all APIs – except for `samples` and `system` endpoints – require the resource to already exist. (In previous versions, a `PATCH` request would create a resource that didn't exist.) To accommodate this change, make a `POST` request to create an item before any `PATCH` request to update it.

Use the Correct API URL

Every Cribl API URL includes a server URL and an endpoint path, which vary depending on the type of deployment.

The server URL follows these patterns:

- Cribl.Cloud deployment:
  `https://<workspaceID>.<organizationID>.cribl.cloud/<endpoint_path>`
- Cribl.Cloud example – default Workspace, "foo-bar-h45h" OrganizationID:
  `https://main.foo-bar-h45h.cribl.cloud/<endpoint_path>`
- On-prem deployment:
  `https://<cribl-stream-leader>:9000/<endpoint_path>`

The endpoint path always begins with `/api/v1/`, but the remainder varies. For example, the `/system/outputs` endpoint follows these patterns:

- Single-instance deployment: `/api/v1/system/outputs/{id}`
- Distributed deployment: `/api/v1/m/{workerGroup}/system/outputs/{id}`
- For licenses limited to a single Fleet, the `{workerGroup}` value will always be `default`

When composing requests to the Cribl API, use the pattern that matches your deployment type. Adapt examples from this page in the same way.

In Cribl.Cloud and other distributed deployments, you must **Commit** and **Deploy** your changes after following the steps in the examples. With the API, you can automate commit/deploy commands, too.

Obtain API Bearer Tokens

Other than calls to `/auth/login` and `/health`, all API requests require a Bearer or access token. You obtain the token in different ways, depending on where your Cribl Edge instance is deployed, as outlined in this section. You can make the listed requests at the command line, programmatically, or using the UI.

On-Prem

To obtain a Bearer token, send a local authentication request and payload to the API as shown below. Adapt the example to include your Leader hostname (or IP address), username, and password.

If you're using SSO/OpenID Connect Authentication, you must toggle **Allow local auth** on, because you'll need to be a Local user when you authenticate via the API.

Request:

```
POST https://<cribl-stream-leader>:9000/api/v1/auth/login
```

Payload:

```
{
    "username": "<username>",
    "password": "<password>"
}
```

Example request using `curl`:

```
curl -X POST https://<cribl-stream-leader>:9000/api/v1/auth/login -H "Content-Type: a
```

You'll get a JSON object as the response, for example:

```
{
    "token": "<JWT_Token>",
    "forcePasswordChange": false
}
```

Use this Bearer token in all subsequent requests. Include it in the Authorization header, like this:

```
Bearer <JWT_token>
```

Cribl.Cloud

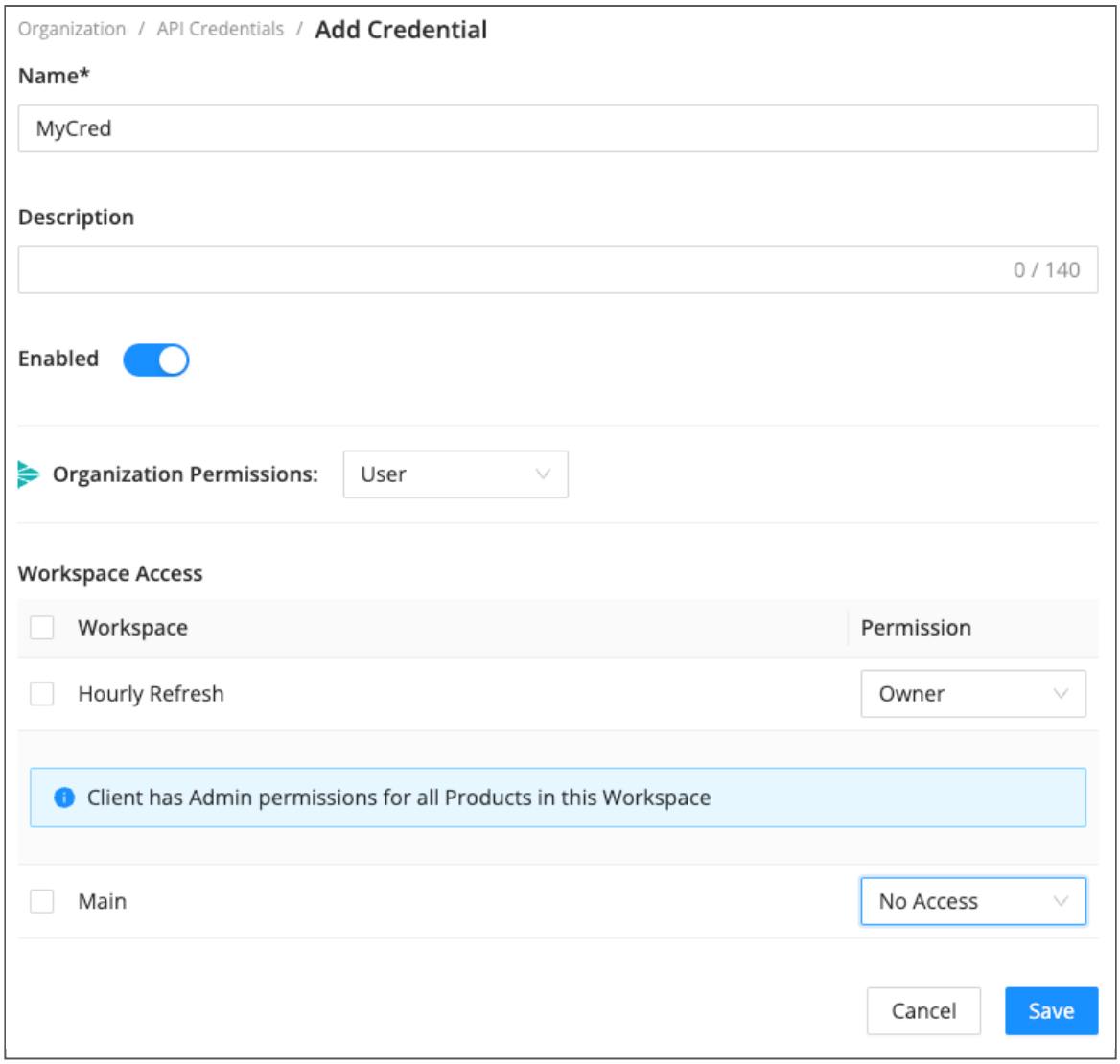
Cribl.Cloud Owners and Admins can generate API access tokens directly in their Organization's portal. This is a two-tier process:

- Generate a Credential to expose a Client ID/Client Secret pair.
- Use the Credential's Client ID and Client Secret to obtain an access token, valid for 24 hours, that can be used to run API calls against Cribl APIs.

Create an API Credential

Owners and Admins can create Credentials:

1. Make sure you are logged in as either an Owner or an Admin.
2. On the top bar, select **Products**, and then select **Cribl**.
3. In the sidebar, expand **Organization**, and then select **API Credentials**.
4. Select **Add Credential** (or **Add an API credential**, if this is the first Credential in the system).
5. Specify a **Name** and an optional **Description** of the Credential's purpose.
6. Under **Organization Permissions**, select the permission that the Credential's tokens will grant.
   The **Organization Permissions** selector is available on certain plan/license tiers. Without a proper license, all tokens will be granted the Admin Role. See Cribl.Cloud Roles for details on the available permissions.
7. If you chose the **User** permission, under **Workspace Access**, define access to specific Workspaces.
   An **Admin** Organization permission automatically grants admin access to all Workspaces.
8. If you selected the **Member** permission for any Workspace, additionally select product-level permissions for that Workspace. (The User permission is not available for API Credentials.)
   An **Admin** Workspace permission automatically grants admin access to all products.
9. Confirm with **Save**.



Creating a Cribl.Cloud API Credential

Use an API Credential

The new API Credential will appear on the **API Credentials** page within a few seconds. Each Credential exposes a **Client ID and Secret**, details of Workspace access and other information. After Credentials are created, Owners and Admins can use them to generate tokens, and then pass these tokens to Cribl APIs.

Getting an API Credential's Client ID and Secret

Obtain Tokens

Obtain a 24-hour access token using a displayed Credential's Client ID and Client Secret. Example curl request, with placeholders:

```
curl --request POST \
--url "https://login.cribl.cloud/oauth/token" \
--header "content-type: application/json" \
--data '{"grant_type":"client_credentials", "client_id": "<client_id>", "client_secr
```

Make API Calls

Using the token returned by the preceding example, you can make requests to Cribl APIs. Example curl request, with placeholder token value:

Single-instance Distributed

```
curl --location 'https://main-<organizationID>.cribl.cloud/api/v1/m/{workerGroup}
--header 'Authorization: Bearer <token-generated-above>'
```

Refresh Tokens

Admins are responsible for ensuring that their applications obtain a new token within each token's 24-hour expiration window. This can be done at any time before expiration.

Manage API Credentials

A Cribl.Cloud Organization's Owners and Admins can view, edit, and disable existing Credentials. Only Owners can delete Credentials.

Update Basic Configurations

You can use the API to programmatically update the configuration of any object type that the API supports (such as Sources and Destinations).

Example: Periodically rotate S3 keys on a preconfigured S3/MinIO destination.

1. Send a `GET` request to the `/outputs` endpoint to retrieve the definition for a Destination (in this case, `MinIO`). The response provides the definition in its payload, as a JSON object.

Example request:

Single-instance | Distributed

```
curl -X GET "<url>/api/v1/m/{workerGroup}/system/inputs/<output id>" -H "accept:
```

Example response:


```
{
  "systemFields": [
    "cribl_pipe"
  ],
  "signatureVersion": "v4",
  "objectACL": "private",
  "partitionExpr": "`${host}/${sourcetype}`",
  "format": "json",
  "baseFileName": "CriblOut",
  "compress": "none",
  "maxFileSizeMB": 32,
  "maxFileOpenTimeSec": 300,
  "maxFileIdleTimeSec": 30,
  "maxOpenFiles": 100,
  "onBackpressure": "block",
  "id": "minio",
  "type": "minio",
  "endpoint": "http://minio:9090",
  "bucket": "test",
  "destPath": "keyprefix",
  "stagePath": "tmp",
  "awsApiKey": "minioadmin",
  "awsSecretKey": "minioadmin"
}
```


2. Edit the definition so that when you send it back in a `PATCH` request, it updates the desired Destination's S3 keys.
    - Edit the value of the `id` field to be the ID of the specific Destination whose keys you want to rotate, for example, `minio_042`.
    - Edit the values of the S3 key fields, for example, `"awsApiKey": "minioadmin_new_api_key"` and `"awsSecretKey": "minioadmin_new_secret_key"`.

3. Send the edited definition as the payload of a `PATCH` request to the `/outputs/{id}` endpoint. This patches (that is, updates) the specified MinIO Destination's configuration.

Upload a Lookup File

This section demonstrates how to upload a Lookup file via the API. The following examples assume that we're uploading the file to a Fleet named `default`.

1. Send a `PUT` request to the `/system/lookups` endpoint to upload the file. Example:

Single-instance Distributed

```
curl -X PUT "<url>/api/v1/m/{workerGroup}/system/lookups?filename=example.csv" \
-H "Authorization: Bearer <token>" -H 'Content-Type: text/csv' \
--data-binary '@/path/to/your/example.csv'
```

You will receive a JSON object response similar to the following example:

```
{"filename":"example.csv.random.tmp","rows":100,"size":200}
```

2. Send a `POST` request referencing the Lookup file to the `/system/lookups` endpoint. Replace the file name with the response from the previous `PUT` request. This both creates the Lookup and moves the Lookup file to its final destination. Example:

Single-instance Distributed

```
curl -X POST "<url>/api/v1/m/{workerGroup}/system/lookups" \
-H "accept: application/json" -H "Authorization: Bearer <token>" \
-H "Content-Type: application/json" \
-d '{"id":"example.csv","fileInfo":{"filename":"example.csv.random.tmp"}}'
```

Example response:

```
{
  "items": [
    {
      "id": "example.csv",
      "size": 200
    }
  ],
  "count": 1
}
```

3. If replacing an existing lookup, send a `PATCH` request referencing the existing file name in the URL and the body. Example:

Single-instance | Distributed

```
curl -X PATCH "<url>/api/v1/m/{workerGroup}/system/lookups/example.csv" \
-H "accept: application/json" -H "Authorization: Bearer <token>" \
-H "Content-Type: application/json" \
-d '{"id":"example.csv","fileInfo":{"filename":"example.csv.random.tmp"}}'
```

Example response:

```
{
  "items": [
    {
      "id": "example.csv",
      "size": 200
    }
  ],
  "count": 1
}
```

Distributed Upload

In a distributed environment, for lookup files of manageable size: Cribl recommends uploading the Lookup file only to the Leader Node, and then making a selective commit and deploy with only those Lookup file changes. The Leader then notifies Edge Nodes that a new configuration is available, and Edge Nodes will pull the new configuration from the Leader Node.

Create HEC Tokens with Python

Cribl SE Jon Rust has written a Python script which demonstrates how to authenticate to the Cribl API, make a simple POST request, and add a new HEC token.

To use the script, you'll need:

- Python 3.
- The Python 3 Requests module (use brew or pip3 to install).
- A working, distributed Cribl Edge installation, with a configured Splunk HEC Source.
- An admin username and password.

The script and instructions for usage can be found in Jon Rust's GitHub repo.

Manage Packs via APIs

You can perform Pack operations by running Cribl API calls on the command line. This is required if you plan to automate Pack operations – for example, in a CI/CD pipeline. For more details, see Managing Packs via API .

15.2. HTTP Response Codes

The following table contains all HTTP response codes that Cribl Edge API can return.

Code	Meaning
200	OK
204	No Content
400	Bad Request
401	Unauthorized
403	Forbidden
404	Not Found
405	Method Not Allowed
415	Unsupported Media Type
500	Internal Server Error
502	Bad Gateway
503	Service Unavailable

15.3. CLI Reference

Command line interface basics

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.

Command Syntax

To execute CLI commands, the basic syntax is:

```
cd $CRIBL_HOME/bin
./cribl <command> <sub-command> <options> <arguments>
```

Not all commands have sub-commands.

To see help for any command, append the `--help` option, for example:

```
./cribl vars --help

./cribl vars get --help

./cribl vars get -i myArray --help
```

The `scope` command is an exception: it has no `--help` option, but it has its own CLI Reference in the AppScope documentation.

Avoiding Surprises

Immediate Execution

As indicated in the sample output below, some commands take effect immediately.

Commands that require further 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, they will be overwritten.

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>`.

`help`

Displays a list of commands with a description (help) for each. Defaults to a selection of generally useful commands.

Usage

```
./cribl help [-a]
```

Options

```
  -a              - Display the list of all commands, except for `scope`.
```

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

auth               - Authentication
boot-start         - Enable/Disable boot-start
diag               - Manage diagnostics bundles
encrypt            - Encrypt with cribl secret
git                - Manage Worker Groups config
keys               - Manage encryption keys
nc                 - Listen on a port for traffic and output stats and data
node               - Execute a JavaScript file
pack               - Manage Cribl Packs
parquet            - Manage Parquet files and schemas
pipe               - Feed stdin to a pipeline
vars               - Manage global variables
```

Starting in version 3.0, former “master” application components have been renamed “Leader.” As long as some legacy terminology remains within CLI commands/options, configuration keys/values, and environment variables, this document will reflect the options available.

limits

Controls the availablity of Cribl features.

Usage

```
./cribl limits <options> <args>
```

Options

```
  [-s <scripts>] - Enable custom scripts API
```

mode-master

Configures instance as a Leader.

Usage

```
./cribl mode-master <options> <args>
```

Options

```
  [-d <deploymentId>]     - Deployment ID for reporting telemetry on multiple deploymer
  [-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 wi
  [-k <privKeyPath>]      - Path on server to the private key to use. PEM format. Can i
  [-c <certPath>]         - Path on server to the certificate to use. PEM format. Can i
  [-u <authToken>]        - Authentication token to include as part of the connection h
  [-S <tls>]              - Sets, or disables, secure communication between Leader and
  [-i <ipWhitelistRegex>] - Regex matching IP addresses that are allowed to establish a
  [-r <resiliency>]       - Resiliency mode for the Leader. Accepts: ["none", "failover
  [-v <failoverVolume>]   - For resiliency=failover, the volume to use as shared storag
```

mode-single

Configures instance as a Single-instance deployment.

Usage

```
./cribl mode-single [--help]
```

Options

```
  [-d <deploymentId>] - Deployment ID for reporting telemetry on multiple deployments
```

mode-edge

Configures Cribl Edge as a Single-instance deployment.

Usage

```
./cribl mode-edge [--help]
```

Options

```
  [-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).
  [-s <socket>]       - Location of the AppScope socket.
```

mode-worker

Configures Cribl Edge as a Worker instance.

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.

Usage

```
./cribl mode-worker -H <host> -p <port> <options> <args>
```

The `-H <host> -p <port>` parameters are required.

Options

```
[-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 (e.g.: 4200).
[-n <certName>]     - Name of the saved certificate to use. Mutually exclusive with
[-k <privKeyPath>]  - Path on server to the private key to use. PEM format. Can refe
[-c <certPath>]     - Path on server to the certificate to use. PEM format. Can refe
[-u <authToken>]    - Authentication token to include as part of the connection head
[-S <tls>]          - Sets, or disables, secure communication between Leader and Worl
[-e <envRegex>]     - Regex to select environment variables to report to Leader. E.g
[-t <tags>]         - Tag values to report to Leader.
[-g <group>]        - Worker Group to report to Leader.
```

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.

mode-managed-edge

Configures Cribl Edge as an 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.

Usage

```
./cribl mode-managed-edge -H <host> -p <port> <options> <args> && ./cribl restart
```

The `-H <host> -p <port>` parameters are required.

Options

```
[-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 (e.g.: 4200).
[-n <certName>]     - Name of the saved certificate to use. Mutually exclusive with
[-k <privKeyPath>]  - Path on server to the private key to use. PEM format. Can refe
[-c <certPath>]     - Path on server to the certificate to use. PEM format. Can refe
[-u <authToken>]    - Authentication token to include as part of the connection head
[-S <tls>]          - Sets, or disables, secure communication between Leader and Worl
[-e <envRegex>]     - Regex to select environment variables to report to Leader. E.g
[-t <tags>]         - Tag values to report to Leader.
[-g <group>]        - Fleet to report to Leader.
```

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.

pack

Manages Cribl Packs.

Usage

```
./cribl pack <sub-command> <options> <args>
```

Sub-commands and Options

```
export            - Export Cribl Packs, args:
   -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.
install           - Install a Cribl Pack, args:
  [-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
list              - List Cribl Packs, args:
  [-v]            - Display all Pack info
  [-g <group>]    - The Worker Group/Fleet to execute within
uninstall         - Uninstall a Cribl Pack, args:
  [-d]            - Run uninstall in debug
  [-g <group>]    - The Worker Group/Fleet to execute within
upgrade           - Upgrade a Cribl Pack, args:
  [-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
```

Sample Response

```
id          version  spec  displayName    author       description
-------------------------------------------------------------------------------
HelloPacks  1.0.0    ----  Hello, Packs!  Cribl, Inc.  A sample pack with a simple e
```

`parquet`

For any Parquet file: view (`cat`) the file, its metadata (`details`), or its `schema`. This command is available only on Linux.

Usage

```
./cribl parquet <sub-command> -f <file>
```

Sub-commands and Options

```
cat             - Cat parquet events to console, args:
   -f <file>    - Path to parquet file
  [-m <max>]    - Maximum events to print
details         - Print all parquet details (schema and statistics), args:
   -f <file>    - Path to parquet file
schema          - Print parquet schema, args:
   -f <file>    - Path to parquet file
```

Examples

View a Parquet file:

```
./cribl parquet cat -f /some_parquet_file
```

```
{"__bytes":274,"Unnamed: 0":0,"name":"apples","quantity":10,"price":2.6,"date":"2018·
...
```

View a Parquet file's metadata, including encoding and compression:

```
./cribl parquet details -f /some_parquet_file
```

```
{
  "schema": {
    "Unnamed: 0": {
      "optional": true,
      "type": "INT64"
    },
    ...
  },
  "path": "../../../src/sluice/js/util/__tests__/data/parquet/single/small_fruits.pa
  "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,
  ...
    }
  ]
}
```


View a Parquet file's schema (not including encoding or compression), expressed as JSON like in the Parquet Schema editor:

```
./cribl parquet schema -f /some_parquet_file
```

```
{
  "Unnamed: 0": {
    "optional": true,
    "type": "INT64"
  },
  "name": {
    "optional": true,
    "type": "STRING"
  },
  "quantity": {
    "optional": true,
    "type": "DOUBLE"
  },
  "price": {
    "optional": true,
    "type": "DOUBLE"
...
  }
}
```


reload

Reloads Cribl Edge. Executes immediately.

Usage

```
./cribl reload [--help]
```

Sample Response

```
Reload request submitted to Cribl
```

restart

Restarts Cribl Edge. Executes immediately.

Executing this command cancels any running collection jobs.

Usage

```
./cribl restart [--help]
```

Sample Response

```
Stopping Cribl, process 17
..............
Cribl is not running
Starting...
....
Cribl started
```

`start`

Starts Cribl Edge. Executes immediately. Upon first run, echoes the Cribl Edge default login credentials.

Usage

```
./cribl start <options> <args>
```

Options

```
[-d <dir>]  - Configuration directory
[-r <role>] - Process role
```

Sample Response

```
Starting...
...
Cribl started
```

`status`

Displays status of Cribl Edge, including the API Server address, instance's mode (Leader or Worker), process ID, and GUID (fictitious example below). Executes immediately.

Usage

```
./cribl status [--help]
```

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
```

`stop`

Stops Cribl Edge. Executes immediately.

Executing this command cancels any running collection jobs.

Usage

```
./cribl stop [--help]
```

Sample Response

```
Stopping Cribl, process 3951
.............
Cribl is not running
```

version

Displays Cribl Edge version. Executes immediately.

Usage

```
./cribl version [--help]
```

Sample Response

```
Software Version: 4.7.3-6f48361f
```

auth

Log into or out of Cribl Edge.

Usage

```
./cribl auth <sub-command> <options> <args>
```

Sub-commands and Options

```
login             - Log in, args:
  [-H <host>]     - Host URL (e.g. http://localhost:9000)
  [-u <username>] - Username
  [-p <password>] - Password
  [-f <file>]     - File with credentials
logout            - Log out
```

Arguments of the `login` subcommand correspond to the following environment variables:

Argument	Variable
<username>	CRIBL_USERNAME
<password>	CRIBL_PASSWORD
<host>	CRIBL_HOST

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>
```

boot-start

Enables or disables Cribl Edge boot-start.

Usage

```
./cribl boot-start <sub-command> <options> <args>
```

Sub-commands and Options

```
  disable             - Disable boot-start, args:
    [-m <manager>]    - Init manager (systemd|initd)
    [-c <configDir>]  - Config directory for the init manager
  enable              - Enable boot-start, args:
    [-m <manager>]    - Init manager (systemd|initd)
    [-u <user>]       - User to run the instance as
    [-c <configDir>]  - Config directory for the init manager
```

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.
```

`cloud-workspace`

Updates an on-premises Cribl Edge Leader with a new config in the `instance.yml` file called `cloudWorkspace`. With this config, the local Cribl Edge Leader will connect to the Cribl.Cloud Leader and send it usage metrics.

Usage

```
./cribl cloud-workspace -H <host> -p <port> # etc: flags
```

Options

```
[-d <deploymentId>]     - Deployment ID used to report telemetry on multiple deploym
[-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 w
[-k <privKeyPath>]      - Path on server to the private key to use. PEM format. Can
[-c <certPath>]         - Path on server to the certificate to use. PEM format. Can
[-u <authToken>]        - Authentication token to include as part of the connection
[-S <tls>]              - Sets, or disables, secure communication between Leader and
[-i <ipWhitelistRegex>] - Regex matching IP addresses that are allowed to establish
[-r <resiliency>]       - Resiliency mode on the Leader. Accepts: ["none", "failover
[-v <failoverVolume>]   - For resiliency=failover, the volume to use as shared storag
```

decrypt

Decrypts data with a secret key. See the related encrypt command.

Usage

```
decrypt -v <encrypted value> -s <path to secret key>
```

Sub-commands and Options

```
 -v <value>             - Value to decrypt.
[-s <criblSecretPath>] - cribl.secret decrypt key path. If you do not specify a crib
```

diag

Manages diagnostic bundles.

Usage

```
./cribl diag <sub-command> <options> <args>
```

Sub-commands and Options

```
create                  - Creates diagnostic bundle for this instance, args:
  [-d]                  - Run create in debug mode
  [-j]                  - Do not append '.txt' to js files
  [-t <maxIncludeJobs>] - Latest number of jobs to include in bundle
  [-M]                  - Exclude metrics from bundle
  [-g]                  - Exclude git from bundle
  [-i]                  - Include install logs
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 c
heapsnapshot            - Generate heap snapshot of a Cribl process, args:
   -p <pid>             - The pid of the process to dump the heap snapshot
```

diag create

```
Created a Cribl diagnostic bundle at /opt/cribl/diag/<product>-zedborcdb72f-20210820
```

diag heapsnapshot -p 12345

The response format is `Heap-<epoch-timestamp>-<pid>.heapsnapshot`:

```
Heap-1672574400000-12345.heapsnapshot
```

encrypt

Encrypts data with a secret key. See the related decrypt command.

Usage

```
encrypt -v my_token -s <path to secret key>
```

Sub-commands and Options

```
 -v <value>             - Value to encrypt.
[-s <criblSecretPath>] - cribl.secret encrypt key path. If you do not specify a crib
```

Sample Response

```
LzmJ3uqA6WXtXjaLPwuM++WX84o3fPxBaubA6Q0look=
```

git

Manages Worker Group/Fleets'/Fleets' configuration.

Usage

```
./cribl git <sub-command> <options> <args>
```

Sub-commands and Options

```
commit           - Commit, args:
  [-g <group>]   - Group ID.
  [-m <message>] - Commit message.
commit-deploy    - Commit and Deploy, args:
   -g <group>    - Group ID.
  [-m <message>] - Commit message.
deploy           - Deploy, args:
   -g <group>    - Group ID.
  [-v <version>] - Deploy version.
list-groups      - List Worker Groups/Fleets.
```

Sample Response

```
Successfully committed version 7c04de1
```

groups

Deprecated. See git.

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`

Usage

```
./cribl keys <sub-command> <options> <args> -g <group>
```

Sub-commands and Options

```
  add                 - Add encryption keys, args:
    [-a <algorithm>] - Encryption algorithm. Supported values: aes-256-cbc (default), a
    [-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 epoch time.
    [-i]             - Use an initialization vector. (IV size configurable for algorith
    [-s <ivSize>]    - (For algo 'aes-256-gcm' only) Initialization vector size (bytes
    [-g <group>]     - Worker Group ID.
  list                - List encryption keys, args:
    [-g <group>]     - Worker Group ID.
```

Sample Response

```
  Adding key succeeded. Key count=1
```

nc

Listens on a port for traffic, and outputs stats and data. (Netcat-like utility.)

Usage

```
./cribl nc -p <port> <options> <args>
```

Options

```
 -p <port>           - Port to listen on.
[-f <family>]        - If this is "6" then nc will listen on ::1, otherwise it listen
[-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
```

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
```

node

Run with no options, displays a command prompt, as shown here:

```
>
```

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:

```
./cribl node -e 'console.log(Date.now())'
1629740667695
```

Usage

```
./cribl node <options> <args>
```

Options

```
[-e <eval>] - String to eval
[-v]        - Prints NodeJS version
```

Sample Response

```
v14.15.1
```

pipe

Feeds stdin to a Pipeline.

Usage

```
./cribl pipe -p <pipelineName> <options> <args>
```

Examples:

```
cat sample.log |  ./cribl pipe -p <pipelineName>
cat sample.log |  ./cribl pipe -p <pipelineName> 2>/dev/null
```

Options

```
 -p <pipeline>    - Pipeline to feed data thru
[-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
```

Sample Response

```
...
{"time":"2021-08-20T20:37:00.017Z","cid":"api","channel":"commands","level":"info","r
{"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","r
{"time":"2021-08-20T20:37:00.028Z","cid":"api","channel":"GrokMgr","level":"info","m
...
```

`scope`

Greps your apps by the syscalls. Executes immediately.

See the AppScope CLI Reference for usage and examples.

`vars`

Manages Cribl Edge Global Variables.

Usage

```
./cribl vars <sub-command> <options> <args>
```

Sub-commands and Options

```
Sub-commands:
add                   - Add global variable, args:
   -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
get                   - List global variables, args:
  [-i <id>]           - Global variable ID
  [-g <group>]        - Group ID
remove                - Remove global variable, args:
   -i <id>            - Global variable ID
  [-g <group>]        - Group ID
update                - Update global variable, args:
   -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
```

Sample Response


```
[
  {
    "type": "number",
    "lib": "cribl",
    "description": "Sample number variable ",
    "value": "42",
    "tags": "cribl,sample",
    "id": "theAnswer"
  }
]
```

15.4. Config Files

Understanding Configuration Paths and Files

Even though all Cribl Edge Routes, Pipelines, and Functions can be managed from the UI, it's important to understand how the configuration works under the hood. Here is how configuration paths and files are laid out on the filesystem.

Path Placeholder	Expanded Path
`$CRIBL_HOME`	Standalone Install: `/path/to/install/cribl/` – referred to below as `$CRIBL_HOME` Cribl App for Splunk Install: `$SPLUNK_HOME/etc/apps/cribl/`

All paths below are relative to `$CRIBL_HOME` in a Single-instance deployment, or to `$CRIBL_HOME/groups/<group-name>/` in a Distributed deployment.

Category	Relative Path
Default Configurations Out-of-the-box defaults (rewritable) and libraries (expandable)	`default/cribl`
Local Configurations User-created integrations and resources	`local/cribl`
System Configuration	`(default|local)/cribl/cribl.yml` See cribl.yml
API Configuration	`(default|local)/cribl/cribl.yml` > `[api]` section See cribl.yml
Source Configuration	`(default|local)/cribl/inputs.yml` See inputs.yml
Destination Configuration	`(default|local)/cribl/outputs.yml` See outputs.yml
License Configuration	`(default|local)/cribl/licenses.yml`

Category	Relative Path
Regexes Configuration	`(default|local)/cribl/regexes.yml`
Breakers Configuration	`(default|local)/cribl/breakers.yml`
Limits Configuration	`(default|local)/cribl/limits.yml`
Service Processes Configuration	`(default|local)/cribl/service.yml` See service.yml
Pipelines Configuration	`(default|local)/cribl/pipelines/<pipeline_name>` Each Pipeline's config resides within its subdirectory.
Packs Configuration	`default/<pack_name>` Each Pack's code and config reside within its subdirectory.
Routes Configuration	`(default|local)/cribl/pipelines/routes.yml`
Functions	`(default|local)/cribl/functions/<function_name>` Each Function's code resides within its subdirectory.
Functions Configuration	`(default|local)/cribl/functions/<function_name>/...` Each Function's config resides within its subdirectory.
Roles Configuration	`(default|local)/cribl/roles.yml` RBAC Role definitions. See roles.yml.
Policies Configuration	`(default|local)/cribl/policies.yml` RBAC Policy definitions. See policies.yml.
Permissions Configuration	`(default|local)/cribl/perms.yml` User permissions. See perms.yml.
Secrets Configuration	`(default|local)/cribl/secrets.yml` Cribl Edge secrets. See secrets.yml.

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 `bin|(local|default)/cribl/...` **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 restarts. Please check current Splunk docs.

Configuration Layering and Precedence

As on most *nix systems, Cribl configurations in `local` take precedence over those in `default`. There is no layering of configuration files.

Editing Configuration Files Manually

When config files **must** be edited manually, save all changes in `local`.

15.4.1. Cribl.Yml

`cribl.yml` contains settings for configuring API and other system properties.

$CRIBL_HOME/default/cribl/cribl.yml

```
api: # [object]
  host: # [string] Address to bind to. Defaults to 0.0.0.0.
  port: # [number] Port to listen to. Defaults to 9000.
  retryCount: # [number] Number of times to retry binding to API port. Defaults to 1
  retrySleepSecs: # [number] Period, in seconds, between consecutive port binding re
  baseUrl: # [string] URL base path from which to serve all assets (useful when behi
  disabled: # [boolean] Flag to enable/disable local UI access. Defaults to false.
  workerRemoteAccess: # [boolean] Enable teleporting to Edge Nodes'. Defaults to fal
  revokeOnRoleChange: # [boolean] Log users out when their roles change. Defaults to
  authTokenTTL: # [number] How long (in seconds) authentication tokens remain valid.
  idleSessionTTL: # [number] How long (in seconds) Cribl Edge will observe no user i
  headers: # [object] Custom HTTP headers to be sent with every response.
  loginRateLimit: # [string] Rate limit, expressed as maximum number of requests per
  ssoRateLimit: # [string] Rate limit for SSO and SLO callback endpoints. Expressed a
  apiCache: # [object] Toggle to 'No' to disable caching of browser's frequent API re
  ssl: # [object] SSL Settings
    disabled: # [boolean] SSL is enabled by default.
    privKeyPath: # [string] Path to private key.
    certPath: # [string] Path to certificate.

auth: # [object] Authentication Settings
  type: # [string] Type - Select one of the supported authentication providers.

  # -------------- if type is ldap ---------------

  secure: # [boolean] Secure - Enable to use a secure ldap connection (ldaps://), dis
  ldapServers: # [array of strings] LDAP servers - List of LDAP servers, each entry s
  bindDN: # [string] Bind DN - Distinguished name of entity to authenticate with LDAP
  bindCredentials: # [string] Password - Distinguished Name password used to authenti
  searchBase: # [string] User search base - Starting point to search LDAP for users,
  usernameField: # [string] Username field - LDAP user search field, e.g. cn or uid
  searchFilter: # [string] User search filter - LDAP search filter to apply when find
  groupSearchBase: # [string] Group search base - Starting point to search LDAP for g
  groupMemberField: # [string] Group member field - LDAP group search field, e.g. mem
  groupSearchFilter: # [string] Group search filter - LDAP search filter to apply whe
  groupField: # [string] Group name field - LDAP group field, e.g. cn
  connectTimeout: # [number] Connection timeout (ms)
  rejectUnauthorized: # [boolean] Reject unauthorized - Valid for secure LDAP connec
  fallback: # [boolean] Fallback on fatal error - Attempt local authentication if LDA
  groups: # [object]
    default: # [string] Default role - Default role assigned to groups not explicitly
    mapping: # [object] Mapping - Group(s)-to-role(s) mappings
```

```
# ------------------------------------------------------


# -------------- if type is saas ---------------

issuer: # [string] Issuer - Issuer from which to accept and validate JWT tokens.
tenantId: # [string] Organization ID - The organization ID within which this instar
loginUrl: # [string] Login URL - The URL to redirect unauthenticated users to.

# ------------------------------------------------------


# -------------- if type is splunk ---------------

host: # [string] Host - Hostname or address of Splunk instance that provides authe
port: # [number] Port - Port of Splunk instance that provides authentication. Defa
ssl: # [boolean] SSL - Whether SSL is enabled on Splunk instance that provides autl
fallback: # [boolean] Fallback on fatal error - Attempt local authentication if Sp
groups: # [object]
  default: # [string] Default role - Default role assigned to groups not explicitl
  mapping: # [object] Mapping - Group(s)-to-role(s) mappings

# ------------------------------------------------------


# -------------- if type is openid ---------------

name: # [string] Provider name - The name of the identity provider service. Manual
audience: # [string] Audience - The Audience from provider configuration. This wil
client_id: # [string] Client ID - The client_id from provider configuration.
client_secret: # [string] Client secret - The client_secret provider configuration
scope: # [string] Scope - Space-separated list of authentication scopes. Default:
auth_url: # [string] Authentication URL - The full path to the provider's authenti
token_url: # [string] Token URL - The full path to the provider's access token URL
userinfo_url: # [string] User Info URL - The full path to the provider's user info
logout_url: # [string] Logout URL - The full path to the provider's logout URL. Le
userIdExpr: # [string] User identifier - Expression used to derive userId from the
rejectUnauthorized: # [boolean] Validate certs - Validate certificates; set to fal
filter_type: # [string] Filter type - Optional method for limiting access per user
groupField: # [string] Group name field - Field on the id_token that contains the
fallback: # [boolean] Allow local auth - Allows locally configured users to log in
groups: # [object]
```

```
    default: # [string] Default role - Default role assigned to groups not explicitly
    mapping: # [object] Mapping - Group(s)-to-role(s) mappings

  # -----------------------------------------------------------

system: # [object]
  upgrade: # [string]
  restart: # [string]
  installType: # [string]

workers: # [object]
  count: # [number]
  memory: # [number]
tls: # [object] Default TLS Settings
  minVersion: # [string] Minimum TLS version - Minimum TLS version. Defaults to TLSv1
  maxVersion: # [string] Maximum TLS version - Maximum TLS version. Defaults to TLSv1
  defaultCipherList: # [string] Default cipher list - Default suite of enabled and di
      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
  defaultEcdhCurve: # [string] ECDH curve - The curve name, or a colon-separated list
  rejectUnauthorized: # [boolean] Validate server certs - Whether to validate server
proxy: # [object]
  useEnvVars: # [boolean]
sni:
```

```
  disableSNIRouting: # [boolean] Manage Server Name Indication (SNI) routing. Set to
git: # [object]
  branch: # [string] Branch - The branch to track in your Stream deployment's git re
  gitOps: # [string] GitOps workflow - The GitOps workflow for managing Stream's con
  commitDeploySingleAction: # [boolean] Collapse actions - When enabled, Commit & De
  defaultCommitMessage: # [string] Default commit message - Enter a default message
  remote: # [string] Remote URL - Enter remote git repo's URL.
  authType: # [string] Authentication type - Git authentication type.

  # -------------- if authType is ssh ---------------

  sshKey: # [string] SSH private key - Enter SSH private key (without passphrase) to
  strictHostKeyChecking: # [boolean] SSH strict host key checking - Validate key aga

  # --------------------------------------------------------


  # -------------- if authType is basic ---------------

  user: # [string] User - Username for authentication.
  password: # [string] Password - Password for authentication. (With GitHub, use a p

  # --------------------------------------------------------

  autoAction: # [string] Scheduled global actions - Global git actions to run automa

  # -------------- if autoAction is commit ---------------

  autoActionSchedule: # [string] Schedule - Cron schedule to run selected git action
  autoActionMessage: # [string] Commit message - Default scheduled commit message.

  # --------------------------------------------------------


  # -------------- if autoAction is push ---------------

  autoActionSchedule: # [string] Schedule - Cron schedule to run selected git action

  # --------------------------------------------------------

  timeout: # [number] Git timeout - Max time (milliseconds) to wait for git processe
```

```
bundler: # [object] - Contains bundler configuration
    bundleGitIgnoredPatterns: # [string] [array of strings] - List of file patterns
```

Example `cribl.yml`:

$CRIBL_HOME/default/cribl/cribl.yml

```
api:
  host: 0.0.0.0
  port: 9000
  retryCount: 120
  retrySleepSecs: 5
  baseUrl: ""
  # Flag to enable/disable UI. Default: false
  disabled : false
  loginRateLimit: 2/second
  ssl:
    disabled: false
    privKeyPath: /path/to/myKey.pem
    certPath: /path/to/myCert.pem
auth:
  type: local
kms.local:
  type: local
crypto:
  keyPath: $CRIBL_HOME/local/cribl/auth/keys.json
system:
  upgrade: api
  restart: api
  installType: standalone
  intercom: true
workers:
  count: -2
  minimum: 2
  memory: 2048
proxy:
  useEnvVars: true
# If there is a custom gitignore rule that excludes cribl.secret from bundle
bundler:
  bundleGitIgnoredPatterns: '**/cribl.secret'
```

15.4.2. Breakers.Yml

Cribl's default Event Breaker Library is stored in `$CRIBL_HOME/default/cribl/breakers.yml`.

$CRIBL_HOME/default/cribl/breakers.yml

```
breaker_id: # [object]
  lib: # [string] Library
  description: # [string] Description - Brief description of this ruleset. Optional.
  tags: # [string] Tags - One or more tags related to this ruleset. Optional.
  rules: # [array] Rules - List of rules. Evaluated in order, top down.
    - name: # [string] Rule Name - Rule Name.
      condition: # [string] Filter Condition - Filter expression (JS) that matches da
      type: # [string] Event Breaker Type - Event Breaker Type
      timestampAnchorRegex: # [string] Timestamp Anchor - Regex to match before atter
      timestamp: # [object] Timestamp Format - Auto, manual format (strptime) or curi
        type: # [string] Timestamp Type
        length: # [number] Length
        format: # [string] Format
      timestampTimezone: # [string] Default timezone - Timezone to assign to timestar
      timestampEarliest: # [string] Earliest timestamp allowed - The earliest timesta
      timestampLatest: # [string] Future timestamp allowed - The latest timestamp val
      maxEventBytes: # [number] Max Event Bytes - The maximum number of bytes that ar
      fields: # [array] Fields - Key value pairs to be added to each event.
      - name: # [string] Name - Field Name.
        value: # [string] Value Expression - JavaScript expression to compute fields
      disabled: # [boolean] Disabled - Allows breaker rule to be enabled or disabled
```

15.4.3. Certificates.Yml

`certificates.yml` maintains a list of configured certificates and their parameters.

$CRIBL_HOME/local/cribl/certificates.yml

```
certificate_id: # [object]
  description: # [string] Description - Brief description of this certificate. Optio
  cert: # [string] Certificate - Drag/drop or upload host certificate, in PEM/Base64
  privKey: # [string] Private key - Certificate private key.
  passphrase: # [string] Passphrase - Passphrase. Optional.
  ca: # [string] CA certificate - Optionally, drag/drop or upload all CA certificate
  inUse: # [array of strings] Referenced - List of configurations referencing this c
```

15.4.4. Groups.Yml

`groups.yml` maintains a list of groups and their configuration versions.

$CRIBL_HOME/local/cribl/groups.yml

```
group_id: # [object]
  configVersion: # [string] Config Version - Configuration version that is active fo
  onPrem: # [boolean] On prem - Whether the group accepts on-prem or cloud worker no
  isFleet: # [boolean] Is Fleet - Fleet groups only manage edge nodes.
  workerRemoteAccess: # [boolean] UI access - Enable authenticated viewing of Worker
```

15.4.5. Inputs.Yml

`inputs.yml` contains settings for configuring inputs into Cribl.

$CRIBL_HOME/default/cribl/inputs.yml

```
inputs: # [object]
  collection_input: # [object]
    type: # [string] Input Type
    breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event b
    staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of tim
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations, o
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.

    # --------------------------------------------------------

    preprocess: # [object]
      disabled: # [boolean] Disabled - Enable Custom Command
      command: # [string] Command - Command to feed the data through (via stdin) and
      args: # [array of strings] Arguments - Arguments
    throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to thrott
    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value, en
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
    environment: # [string] Environment - Optionally, enable this config only on a s
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesyste
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
      commitFrequency: # [number] Commit frequency - The number of events to send do
      maxFileSize: # [string] Max file size - The maximum size to store in each queu
      maxSize: # [string] Max queue size - The maximum amount of disk space the queu
      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 - Add tags for filtering and grouping in S
```

```
kafka_input: # [object]
  type: # [string] Input Type
  brokers: # [array of strings] Brokers - List of Kafka brokers to use, eg. localh
  topics: # [array of strings] [required] Topic - Topic to subscribe to. Warning:
  groupId: # [string] Group ID - Specifies the consumer group this instance belong
  fromBeginning: # [boolean] From beginning - Whether to start reading from earlie
  kafkaSchemaRegistry: # [object] Kafka Schema Registry Authentication
    disabled: # [boolean] Disabled - Enable Schema Registry

    # -------------- if disabled is false ---------------

    schemaRegistryURL: # [string] Schema Registry URL - URL for access to the Conf
    tls: # [object] TLS settings (client side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      rejectUnauthorized: # [boolean] Validate server certs - Reject certs that ar
      servername: # [string] Server name (SNI) - Server name for the SNI (Server Na
      certificateName: # [string] Certificate name - The name of the predefined ce
      caPath: # [string] CA certificate path - Path on client in which to find CA
      privKeyPath: # [string] Private key path (mutual auth) - Path on client in wh
      certPath: # [string] Certificate path (mutual auth) - Path on client in whic
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
      minVersion: # [string] Minimum TLS version - Minimum TLS version to use when
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when

      # ----------------------------------------------------------



    # ----------------------------------------------------------

  connectionTimeout: # [number] Connection timeout (ms) - Maximum time to wait for
  requestTimeout: # [number] Request timeout (ms) - Maximum time to wait for a suc
  sasl: # [object] Authentication - Authentication parameters to use when connecti
    disabled: # [boolean] Disabled - Enable Authentication

    # -------------- if disabled is false ---------------

    mechanism: # [string] SASL mechanism - SASL authentication mechanism to use.

    # ----------------------------------------------------------
```

```
tls: # [object] TLS settings (client side)
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
  servername: # [string] Server name (SNI) - Server name for the SNI (Server Nam
  certificateName: # [string] Certificate name - The name of the predefined cert
  caPath: # [string] CA certificate path - Path on client in which to find CA ce
  privKeyPath: # [string] Private key path (mutual auth) - Path on client in whi
  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 - Minimum TLS version to use when c
  maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when c

  # ------------------------------------------------------

sessionTimeout: # [number] Session timeout (ms) -
  Timeout used to detect client failures when using Kafka's group management fac
  If the client sends the broker no heartbeats before this timeout expires,
  the broker will remove this client from the group, and will initiate a rebalan
  Value must be between the broker's configured group.min.session.timeout.ms and
  See details [here](https://kafka.apache.org/documentation/#consumerconfigs_ses
rebalanceTimeout: # [number] Rebalance timeout (ms) -
  Maximum allowed time for each worker to join the group after a rebalance has b
  If the timeout is exceeded, the coordinator broker will remove the worker from
  See details [here](https://kafka.apache.org/documentation/#connectconfigs_reba
heartbeatInterval: # [number] Heartbeat interval (ms) -
  Expected time between heartbeats to the consumer coordinator when using Kafka'
  Value must be lower than sessionTimeout, and typically should not exceed 1/3 o
  See details [here](https://kafka.apache.org/documentation/#consumerconfigs_hea
metadata: # [array] Fields - Fields to add to events from this input.
  - name: # [string] Name - Field name
    value: # [string] Value - JavaScript expression to compute field's value, en
disabled: # [boolean] Disabled - Enable/disable this input
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

# -------------- if sendToRoutes is false ---------------

connections: # [array] Quick Connections - Direct connections to Destinations, o
  - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
    output: # [string] Destination - Select a Destination.
```

```
  # -------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a sp
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyster
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
    commitFrequency: # [number] Commit frequency - The number of events to send dow
    maxFileSize: # [string] Max file size - The maximum size to store in each queu
    maxSize: # [string] Max queue size - The maximum amount of disk space the queu
    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 - Add tags for filtering and grouping in S
http_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled - Enable/disable this input
  host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses
  port: # [number] [required] Port - Port to listen to.
  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 name - The name of the predefined cert
    privKeyPath: # [string] Private key path - Path on server containing the priva
    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) - Whether to requir
    minVersion: # [string] Minimum TLS version - Minimum TLS version to accept fro
    maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept fro

    # -------------------------------------------------------

  maxActiveReq: # [number] Max active requests - Maximum number of active requests
```

```
  enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
  captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add
  activityLogSampleRate: # [number] Activity log sample rate - How often request a
  requestTimeout: # [number] Request timeout (seconds) - How long to wait for an i
  criblAPI: # [string] Cribl HTTP Event API - Absolute path on which to listen for
  elasticAPI: # [string] Elasticsearch API endpoint (Bulk API) - Absolute path on
  splunkHecAPI: # [string] Splunk HEC Endpoint - Absolute path on which listen for
  splunkHecAcks: # [boolean] Splunk HEC Acks - Whether to enable Splunk HEC acknow
  metadata: # [array] Fields - Fields to add to events from this input.
    - name: # [string] Name - Field name
      value: # [string] Value - JavaScript expression to compute field's value, en
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Quick Connections - Direct connections to Destinations, o
    - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
      output: # [string] Destination - Select a Destination.

  # --------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a s
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyste
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
    commitFrequency: # [number] Commit frequency - The number of events to send do
    maxFileSize: # [string] Max file size - The maximum size to store in each queu
    maxSize: # [string] Max queue size - The maximum amount of disk space the queu
    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 - Add tags for filtering and grouping in S
splunk_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled - Enable/disable this input
  host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresse
```

```
port: # [number] [required] Port - Port to listen to.
tls: # [object] TLS settings (server side)
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  certificateName: # [string] Certificate name - The name of the predefined certi
  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) - Whether to require
  minVersion: # [string] Minimum TLS version - Minimum TLS version to accept from
  maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept from

  # ----------------------------------------------------

ipWhitelistRegex: # [string] IP Allowlist Regex - Regex matching IP addresses tha
maxActiveCxn: # [number] Max Active Connections - Maximum number of active connec
enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection i
metadata: # [array] Fields - Fields to add to events from this input.
  - name: # [string] Name - 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] Event Breaker buffer timeout - The amount of time
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 - Optional token description
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly f

# -------------- if sendToRoutes is false ---------------

connections: # [array] Quick Connections - Direct connections to Destinations, op
  - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
    output: # [string] Destination - Select a Destination.

# ----------------------------------------------------

environment: # [string] Environment - Optionally, enable this config only on a sp
pqEnabled: # [boolean] Enable Persistent Queue

# -------------- if pqEnabled is true ---------------
```

```
  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyste
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
    commitFrequency: # [number] Commit frequency - The number of events to send do
    maxFileSize: # [string] Max file size - The maximum size to store in each queu
    maxSize: # [string] Max queue size - The maximum amount of disk space the queu
    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 - Add tags for filtering and grouping in S
splunk_search_input: # [object]
  searchHead: # [string] Search head - Search head base URL, can be expression, de
  search: # [string] [required] Search - Enter Splunk search here. For example: 'i
  earliest: # [string] Earliest - The earliest time boundary for the search. Can b
  latest: # [string] Latest - The latest time boundary for the search. Can be an e
  cronSchedule: # [string] [required] Cron schedule - A cron schedule on which to
  endpoint: # [string] [required] Search endpoint - REST API used to create a sear
  outputMode: # [string] [required] Output mode - Format of the returned output
  endpointParams: # [array] Endpoint parameters - Optional request parameters to s
    - name: # [string] Name - Parameter name
      value: # [string] Value - JavaScript expression to compute the parameter's v
  endpointHeaders: # [array] Endpoint headers - Optional request headers to send t
    - name: # [string] Name - Header Name
      value: # [string] Value - JavaScript expression to compute the header's valu
  logLevel: # [string] Log level - Collector runtime log Level (verbosity).
  requestTimeout: # [number] Request timeout (seconds) - HTTP request inactivity t
  useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS lo
  keepAliveTime: # [number] Keep Alive Time (seconds) - How often workers should c
  maxMissedKeepAlives: # [number] Worker Timeout (periods) - The number of Keep Al
  metadata: # [array] Fields - Fields to add to events from this input.
    - name: # [string] Name - Field name
      value: # [string] Value - JavaScript expression to compute field's value, en
  breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event b
  staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of tim
  authType: # [string] Authentication type - Splunk Search authentication type

  # -------------- if authType is basic ---------------

  username: # [string] Username - Username for Basic authentication
  password: # [string] Password - Password for Basic authentication
```

```
# ------------------------------------------------------


# -------------- 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 tha

# ------------------------------------------------------


# -------------- if authType is textSecret ---------------

textSecret: # [string] Token (text secret) - Select or create a stored text secr

# ------------------------------------------------------

type: # [string] Input Type
disabled: # [boolean] Disabled - Enable/disable this input
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

# -------------- if sendToRoutes is false ---------------

connections: # [array] Quick Connections - Direct connections to Destinations, o
  - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
    output: # [string] Destination - Select a Destination.

# ------------------------------------------------------

environment: # [string] Environment - Optionally, enable this config only on a s
pqEnabled: # [boolean] Enable Persistent Queue

# -------------- if pqEnabled is true ---------------

pq: # [object]
```

```
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyster
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
    commitFrequency: # [number] Commit frequency - The number of events to send dow
    maxFileSize: # [string] Max file size - The maximum size to store in each queu
    maxSize: # [string] Max queue size - The maximum amount of disk space the queu
    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 - Add tags for filtering and grouping in S
splunk_hec_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled - Enable/disable this input
  host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresse
  port: # [number] [required] Port - Port to listen to.
  authTokens: # [array] Auth tokens - Shared secrets to be provided by any client
    - token: # [string] Token - Shared secret to be provided by any client (Author
      description: # [string] Description - Optional token description
      metadata: # [array] Fields - Fields to add to events referencing this token.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
  tls: # [object] TLS settings (server side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    certificateName: # [string] Certificate name - The name of the predefined cert
    privKeyPath: # [string] Private key path - Path on server containing the priva
    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 certific
    requestCert: # [boolean] Authenticate client (mutual auth) - Whether to requir
    minVersion: # [string] Minimum TLS version - Minimum TLS version to accept fro
    maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept fro

    # -------------------------------------------------------

  maxActiveReq: # [number] Max active requests - Maximum number of active requests
  enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
  captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add
  activityLogSampleRate: # [number] Activity log sample rate - How often request a
  requestTimeout: # [number] Request timeout (seconds) - How long to wait for an i
```

```
  splunkHecAPI: # [string] [required] Splunk HEC Endpoint - Absolute path on which
  metadata: # [array] Fields - Fields to add to every event. May be overridden by
    - name: # [string] Name - Field name
      value: # [string] Value - JavaScript expression to compute field's value, en
  allowedIndexes: # [array of strings] Allowed Indexes - List values allowed in HE
  splunkHecAcks: # [boolean] Splunk HEC Acks - Whether to enable Splunk HEC acknow
  breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event b
  staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of tim
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Quick Connections - Direct connections to Destinations, o
    - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
      output: # [string] Destination - Select a Destination.

  # --------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a s
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyste
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
    commitFrequency: # [number] Commit frequency - The number of events to send do
    maxFileSize: # [string] Max file size - The maximum size to store in each queu
    maxSize: # [string] Max queue size - The maximum amount of disk space the queu
    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 - Add tags for filtering and grouping in S
azure_blob_input: # [object]
  type: # [string] Input Type
  queueName: # [string] Queue - The storage account queue name blob notifications
  fileFilter: # [string] Filename filter - Regex matching file names to download a
  visibilityTimeout: # [number] Visibility timeout (secs) - The duration (in secon
  numReceivers: # [number] Num receivers - The Number of receiver processes to run
  maxMessages: # [number] Max messages - The maximum number of messages to return
```

```
servicePeriodSecs: # [number] Service period (secs) - The duration (in seconds) u
skipOnError: # [boolean] Skip file on error - Toggle to Yes to skip files that t
metadata: # [array] Fields - Fields to add to events from this input.
  - name: # [string] Name - Field name
    value: # [string] Value - JavaScript expression to compute field's value, en
breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event b
staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of tim
parquetChunkSizeMB: # [number] Max Parquet chunk size (MB) - Maximum file size f
parquetChunkDownloadTimeout: # [number] Parquet chunk download timeout (seconds)
authType: # [string] Authentication method - Enter connection string directly, o
connectionString: # [string] Connection string - Enter your Azure Storage accoun

# -------------- if authType is manual ---------------


# -----------------------------------------------------

textSecret: # [string] Connection string (text secret) - Select or create a stor

# -------------- if authType is secret ---------------


# -----------------------------------------------------

disabled: # [boolean] Disabled - Enable/disable this input
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

# -------------- if sendToRoutes is false ---------------

connections: # [array] Quick Connections - Direct connections to Destinations, o
  - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
    output: # [string] Destination - Select a Destination.

# -----------------------------------------------------

environment: # [string] Environment - Optionally, enable this config only on a s
pqEnabled: # [boolean] Enable Persistent Queue

# -------------- if pqEnabled is true ---------------

pq: # [object]
  mode: # [string] Mode - With Smart mode, PQ will write events to the filesyste
```

```
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
    commitFrequency: # [number] Commit frequency - The number of events to send do
    maxFileSize: # [string] Max file size - The maximum size to store in each queu
    maxSize: # [string] Max queue size - The maximum amount of disk space the queu
    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 - Add tags for filtering and grouping in S
elastic_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled - Enable/disable this input
  host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresse
  port: # [number] [required] Port - Port to listen to.
  tls: # [object] TLS settings (server side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    certificateName: # [string] Certificate name - The name of the predefined cert
    privKeyPath: # [string] Private key path - Path on server containing the priva
    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) - Whether to requir
    minVersion: # [string] Minimum TLS version - Minimum TLS version to accept fro
    maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept fro

    # ------------------------------------------------------

  maxActiveReq: # [number] Max active requests - Maximum number of active requests
  enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
  captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add
  activityLogSampleRate: # [number] Activity log sample rate - How often request a
  requestTimeout: # [number] Request timeout (seconds) - How long to wait for an i
  elasticAPI: # [string] [required] Elasticsearch API endpoint - Absolute path on
  authType: # [string] Authentication type - Elastic authentication type

  # -------------- if authType is basic ---------------

  username: # [string] Username - Username for Basic authentication
  password: # [string] Password - Password for Basic authentication
```

```
# ------------------------------------------------------

# -------------- if authType is credentialsSecret ---------------

credentialsSecret: # [string] Credentials secret - Select or create a secret tha

# ------------------------------------------------------

# -------------- if authType is authTokens ---------------

authTokens: # [array of strings] Token - Bearer tokens to include in the authori

# ------------------------------------------------------

apiVersion: # [string] API Version - The API version to use for communicating wi

# -------------- if apiVersion is custom ---------------

customAPIVersion: # [string] Custom API Version - Custom version information to

# ------------------------------------------------------

extraHttpHeaders: # [array] Extra HTTP headers - Extra HTTP headers.
  - name: # [string] Name - Field name
    value: # [string] Value - Field value
metadata: # [array] Fields - Fields to add to events from this input.
  - name: # [string] Name - Field name
    value: # [string] Value - JavaScript expression to compute field's value, en
proxyMode: # [object]
  enabled: # [boolean] Enable Proxy Mode - Enable proxying of non-bulk API reque

  # -------------- if enabled is true ---------------

  url: # [string] Proxy URL - URL of the Elastic server to proxy non-bulk reques
  removeHeaders: # [array of strings] Remove headers - List of headers to remove
  timeoutSec: # [number] Proxy request timeout - Amount of time, in seconds, to
  authType: # [string] Authentication method - Enter credentials directly, or se

  # ------------------------------------------------------
```

```
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Quick Connections - Direct connections to Destinations, o
    - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
      output: # [string] Destination - Select a Destination.

  # -------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a s
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyste
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
    commitFrequency: # [number] Commit frequency - The number of events to send do
    maxFileSize: # [string] Max file size - The maximum size to store in each queu
    maxSize: # [string] Max queue size - The maximum amount of disk space the queu
    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 - Add tags for filtering and grouping in S
confluent_cloud_input: # [object]
  type: # [string] Input Type
  brokers: # [array of strings] Brokers - List of Confluent Cloud brokers to use,
  tls: # [object] TLS settings (client side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
    servername: # [string] Server name (SNI) - Server name for the SNI (Server Nam
    certificateName: # [string] Certificate name - The name of the predefined cert
    caPath: # [string] CA certificate path - Path on client in which to find CA ce
    privKeyPath: # [string] Private key path (mutual auth) - Path on client in whi
    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 - Minimum TLS version to use when co
  maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when co

  # ------------------------------------------------------

topics: # [array of strings] [required] Topic - Topic to subscribe to. Warning: T
groupId: # [string] Group ID - Specifies the consumer group this instance belongs
fromBeginning: # [boolean] From beginning - Whether to start reading from earlies
kafkaSchemaRegistry: # [object] Kafka Schema Registry Authentication
  disabled: # [boolean] Disabled - Enable Schema Registry

  # -------------- if disabled is false ---------------

  schemaRegistryURL: # [string] Schema Registry URL - URL for access to the Confl
  tls: # [object] TLS settings (client side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
    servername: # [string] Server name (SNI) - Server name for the SNI (Server Na
    certificateName: # [string] Certificate name - The name of the predefined cer
    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 - Minimum TLS version to use when
    maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when

    # ------------------------------------------------------


  # ------------------------------------------------------

connectionTimeout: # [number] Connection timeout (ms) - Maximum time to wait for
requestTimeout: # [number] Request timeout (ms) - Maximum time to wait for a suc
sasl: # [object] Authentication - Authentication parameters to use when connecti
  disabled: # [boolean] Disabled - Enable Authentication

  # -------------- if disabled is false ---------------

  mechanism: # [string] SASL mechanism - SASL authentication mechanism to use.
```

```
  # ------------------------------------------------------

sessionTimeout: # [number] Session timeout (ms) -
  Timeout used to detect client failures when using Kafka's group management fac
  If the client sends the broker no heartbeats before this timeout expires,
  the broker will remove this client from the group, and will initiate a rebalan
  Value must be between the broker's configured group.min.session.timeout.ms and
  See details [here](https://kafka.apache.org/documentation/#consumerconfigs_ses
rebalanceTimeout: # [number] Rebalance timeout (ms) -
  Maximum allowed time for each worker to join the group after a rebalance has b
  If the timeout is exceeded, the coordinator broker will remove the worker from
  See details [here](https://kafka.apache.org/documentation/#connectconfigs_reba
heartbeatInterval: # [number] Heartbeat interval (ms) -
  Expected time between heartbeats to the consumer coordinator when using Kafka'
  Value must be lower than sessionTimeout, and typically should not exceed 1/3 o
  See details [here](https://kafka.apache.org/documentation/#consumerconfigs_hea
metadata: # [array] Fields - Fields to add to events from this input.
  - name: # [string] Name - Field name
    value: # [string] Value - JavaScript expression to compute field's value, en
disabled: # [boolean] Disabled - Enable/disable this input
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

# -------------- if sendToRoutes is false ---------------

connections: # [array] Quick Connections - Direct connections to Destinations, o
  - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
    output: # [string] Destination - Select a Destination.

# -------------------------------------------------------

environment: # [string] Environment - Optionally, enable this config only on a s
pqEnabled: # [boolean] Enable Persistent Queue

# -------------- if pqEnabled is true ---------------

pq: # [object]
  mode: # [string] Mode - With Smart mode, PQ will write events to the filesyste
  maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
  commitFrequency: # [number] Commit frequency - The number of events to send do
  maxFileSize: # [string] Max file size - The maximum size to store in each queu
  maxSize: # [string] Max queue size - The maximum amount of disk space the queu
  path: # [string] Queue file path - The location for the persistent queue files
```

```
    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 - Add tags for filtering and grouping in S
grafana_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled - Enable/disable this input
  host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses
  port: # [number] [required] Port - Port to listen to.
  tls: # [object] TLS settings (server side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    certificateName: # [string] Certificate name - The name of the predefined certi
    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) - Whether to requir
    minVersion: # [string] Minimum TLS version - Minimum TLS version to accept fro
    maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept fro

    # ------------------------------------------------------

  maxActiveReq: # [number] Max active requests - Maximum number of active requests
  enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection i
  captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add i
  activityLogSampleRate: # [number] Activity log sample rate - How often request ac
  requestTimeout: # [number] Request timeout (seconds) - How long to wait for an in
  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
  keepAliveTimeout: # [number] Keep alive timeout (seconds) - Maximum time to wait
  prometheusAuth: # [object]
    authType: # [string] Authentication type - Remote Write authentication type

    # -------------- if authType is basic ---------------

    username: # [string] Username - Username for Basic authentication
    password: # [string] Password - Password for Basic authentication

    # ------------------------------------------------------
```

```
    "

    # -------------- 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 tl

    # ----------------------------------------------------


    # -------------- if authType is textSecret ---------------

    textSecret: # [string] Token (text secret) - Select or create a stored text se

    # ----------------------------------------------------

  lokiAuth: # [object]
    authType: # [string] Authentication type - Loki logs authentication type

    # -------------- if authType is basic ---------------

    username: # [string] Username - Username for Basic authentication
    password: # [string] Password - Password for Basic authentication

    # ----------------------------------------------------


    # -------------- 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 tl
```

```
    credentialsSecret: # [string] Credentials Secret - Select or create a secret t

    # -------------------------------------------------------


    # -------------- if authType is textSecret ---------------

    textSecret: # [string] Token (text secret) - Select or create a stored text se

    # -------------------------------------------------------

  metadata: # [array] Fields - Fields to add to events from this input.
    - name: # [string] Name - Field name
      value: # [string] Value - JavaScript expression to compute field's value, en
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Quick Connections - Direct connections to Destinations, o
    - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
      output: # [string] Destination - Select a Destination.

  # -------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a s
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyste
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
    commitFrequency: # [number] Commit frequency - The number of events to send do
    maxFileSize: # [string] Max file size - The maximum size to store in each queu
    maxSize: # [string] Max queue size - The maximum amount of disk space the queu
    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 - Add tags for filtering and grouping in S
loki_input: # [object]
  type: # [string] Input Type
```

```
type: # [string] Input type
disabled: # [boolean] Disabled - Enable/disable this input
host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses
port: # [number] [required] Port - Port to listen to.
tls: # [object] TLS settings (server side)
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  certificateName: # [string] Certificate name - The name of the predefined certi
  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) - Whether to require
  minVersion: # [string] Minimum TLS version - Minimum TLS version to accept from
  maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept from

  # ----------------------------------------------------------

maxActiveReq: # [number] Max active requests - Maximum number of active requests
enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection i
captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add i
activityLogSampleRate: # [number] Activity log sample rate - How often request ac
requestTimeout: # [number] Request timeout (seconds) - How long to wait for an in
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 - Username for Basic authentication
password: # [string] Password - Password for Basic authentication

# ----------------------------------------------------------


# -------------- if authType is token ---------------

token: # [string] Token - Bearer token to include in the authorization header

# ----------------------------------------------------------


#                if authType is credentialSecret
```

```
# -------------- if authType is credentialsSecret ---------------

credentialsSecret: # [string] Credentials secret - Select or create a secret tha

# ----------------------------------------------------------


# -------------- if authType is textSecret ---------------

textSecret: # [string] Token (text secret) - Select or create a stored text secr

# ----------------------------------------------------------

metadata: # [array] Fields - Fields to add to events from this input.
  - name: # [string] Name - Field name
    value: # [string] Value - JavaScript expression to compute field's value, en
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

# -------------- if sendToRoutes is false ---------------

connections: # [array] Quick Connections - Direct connections to Destinations, o
  - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
    output: # [string] Destination - Select a Destination.

# ----------------------------------------------------------

environment: # [string] Environment - Optionally, enable this config only on a s
pqEnabled: # [boolean] Enable Persistent Queue

# -------------- if pqEnabled is true ---------------

pq: # [object]
  mode: # [string] Mode - With Smart mode, PQ will write events to the filesyste
  maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
  commitFrequency: # [number] Commit frequency - The number of events to send do
  maxFileSize: # [string] Max file size - The maximum size to store in each queu
  maxSize: # [string] Max queue size - The maximum amount of disk space the queu
  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 - Add tags for filtering and grouping in S
```

```
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in S
prometheus_rw_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled - Enable/disable this input
  host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses
  port: # [number] [required] Port - Port to listen to.
  tls: # [object] TLS settings (server side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    certificateName: # [string] Certificate name - The name of the predefined certi
    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) - Whether to require
    minVersion: # [string] Minimum TLS version - Minimum TLS version to accept from
    maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept from

    # ----------------------------------------------------------

  maxActiveReq: # [number] Max active requests - Maximum number of active requests
  enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection i
  captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add i
  activityLogSampleRate: # [number] Activity log sample rate - How often request ac
  requestTimeout: # [number] Request timeout (seconds) - How long to wait for an in
  prometheusAPI: # [string] [required] Remote Write API endpoint - Absolute path on
  keepAliveTimeout: # [number] Keep alive timeout (seconds) - Maximum time to wait
  authType: # [string] Authentication type - Remote Write authentication type

  # -------------- if authType is basic ---------------

  username: # [string] Username - Username for Basic authentication
  password: # [string] Password - Password for Basic authentication

  # ----------------------------------------------------------


  # -------------- 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 tha

# ------------------------------------------------------


# -------------- if authType is textSecret ---------------

textSecret: # [string] Token (text secret) - Select or create a stored text secr

# ------------------------------------------------------

metadata: # [array] Fields - Fields to add to events from this input.
  - name: # [string] Name - Field name
    value: # [string] Value - JavaScript expression to compute field's value, en
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

# -------------- if sendToRoutes is false ---------------

connections: # [array] Quick Connections - Direct connections to Destinations, o
  - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
    output: # [string] Destination - Select a Destination.

# ------------------------------------------------------

environment: # [string] Environment - Optionally, enable this config only on a s
pqEnabled: # [boolean] Enable Persistent Queue

# -------------- if pqEnabled is true ---------------

pq: # [object]
  mode: # [string] Mode - With Smart mode, PQ will write events to the filesyste
  maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
  commitFrequency: # [number] Commit frequency - The number of events to send do
  maxFileSize: # [string] Max file size - The maximum size to store in each queu
  maxSize: # [string] Max queue size - The maximum amount of disk space the queu
  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 - Add tags for filtering and grouping in S
prometheus_input: # [object]
  dimensionList: # [array of strings] Extra Dimensions - Other dimensions to inclu
  discoveryType: # [string] Discovery Type - Target discovery mechanism. Use stati

  # -------------- if discoveryType is static ---------------

  targetList: # [array of strings] Targets - List of Prometheus targets to pull me

  # ------------------------------------------------------


  # -------------- 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 me
  scrapePath: # [string] Metrics Path - Path to use when collecting metrics from d

  # ------------------------------------------------------


  # -------------- if discoveryType is ec2 ---------------

  usePublicIp: # [boolean] Use Public IP - Use public IP address for discovered ta
  scrapeProtocol: # [string] Metrics Protocol - Protocol to use when collecting me
  scrapePort: # [number] Metrics Port - The port number in the metrics URL for dis
  scrapePath: # [string] Metrics Path - Path to use when collecting metrics from d
  searchFilter: # [array] Search Filter - EC2 Instance Search Filter
    - Name: # [string] Filter Name - Search filter attribute name, see: https://do
      Values: # [array of strings] Filter Values - Search Filter Values, if empty
  awsAuthenticationMethod: # [string] Authentication method - AWS authentication m
  awsSecretKey: # [string] Secret key - Secret key
  region: # [string] Region - Region where the EC2 is located
  endpoint: # [string] Endpoint - EC2 service endpoint. If empty, defaults to AWS'
  signatureVersion: # [string] Signature version - Signature version to use for si
  reuseConnections: # [boolean] Reuse connections - Whether to reuse connections b
  rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to re
  enableAssumeRole: # [boolean] Enable for EC2 - Use Assume Role credentials to ac
  assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the rol
```

```
assumeRoleExternalId: # [string] External ID - External ID to use when assuming :

# ------------------------------------------------------

interval: # [number] Poll Interval - How often in minutes to scrape targets for r
logLevel: # [string] [required] Log Level - Collector runtime Log Level
keepAliveTime: # [number] Keep Alive Time (seconds) - How often workers should cl
maxMissedKeepAlives: # [number] Worker Timeout (periods) - The number of Keep Al:
metadata: # [array] Fields - Fields to add to events from this input.
  - name: # [string] Name - Field name
    value: # [string] Value - JavaScript expression to compute field's value, en
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 tha

# -------------- if authType is secret ---------------


# ------------------------------------------------------

type: # [string] Input Type
disabled: # [boolean] Disabled - Enable/disable this input
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

# -------------- if sendToRoutes is false ---------------

connections: # [array] Quick Connections - Direct connections to Destinations, o
  - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
    output: # [string] Destination - Select a Destination.

# ------------------------------------------------------

environment: # [string] Environment - Optionally, enable this config only on a s
pqEnabled: # [boolean] Enable Persistent Queue
```

```
  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyste
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
    commitFrequency: # [number] Commit frequency - The number of events to send do
    maxFileSize: # [string] Max file size - The maximum size to store in each queu
    maxSize: # [string] Max queue size - The maximum amount of disk space the queu
    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 - Add tags for filtering and grouping in S
office365_mgmt_input: # [object]
  type: # [string] Input Type
  tenantId: # [string] Tenant ID - Office 365 Azure Tenant ID
  appId: # [string] [required] App ID - Office 365 Azure Application ID
  timeout: # [number] Timeout (secs) - HTTP request inactivity timeout, use 0 to d
  keepAliveTime: # [number] Keep Alive Time (seconds) - How often workers should c
  maxMissedKeepAlives: # [number] Worker Timeout (periods) - The number of Keep Al
  metadata: # [array] Fields - Fields to add to events from this input.
    - name: # [string] Name - Field name
      value: # [string] Value - JavaScript expression to compute field's value, en
  planType: # [string] [required] Subscription Plan - Office 365 subscription plan
  publisherIdentifier: # [string] Publisher Identifier - Optional Publisher Identi
  contentConfig: # [array] Content Types - Enable Office 365 Management Activity A
    - contentType: # [string] Content Type - Office 365 Management Activity API Co
      description: # [string] Interval Description - If interval type is minutes t
      interval: # [number] Interval
      logLevel: # [string] Log Level - Collector runtime Log Level
      enabled: # [boolean] Enabled
  authType: # [string] Authentication method - Enter client secret directly, or se
  clientSecret: # [string] Client secret - Office 365 Azure client secret

  # -------------- if authType is manual ---------------


  # ----------------------------------------------------------

  textSecret: # [string] Client secret (text secret) - Select or create a stored t

  # -------------- if authType is secret ---------------
```

```
  # -------------------------------------------------------

  disabled: # [boolean] Disabled - Enable/disable this input
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Quick Connections - Direct connections to Destinations, o
    - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
      output: # [string] Destination - Select a Destination.

  # -------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a s
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyste
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
    commitFrequency: # [number] Commit frequency - The number of events to send do
    maxFileSize: # [string] Max file size - The maximum size to store in each queu
    maxSize: # [string] Max queue size - The maximum amount of disk space the queu
    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 - Add tags for filtering and grouping in S
office365_service_input: # [object]
  type: # [string] Input Type
  tenantId: # [string] Tenant ID - Office 365 Azure Tenant ID
  appId: # [string] [required] App ID - Office 365 Azure Application ID
  timeout: # [number] Timeout (secs) - HTTP request inactivity timeout, use 0 to d
  keepAliveTime: # [number] Keep Alive Time (seconds) - How often workers should c
  maxMissedKeepAlives: # [number] Worker Timeout (periods) - The number of Keep Al
  metadata: # [array] Fields - Fields to add to events from this input.
    - name: # [string] Name - Field name
      value: # [string] Value - JavaScript expression to compute field's value, en
```

```
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 tl
    interval: # [number] Interval
    logLevel: # [string] Log Level - Collector runtime Log Level
    enabled: # [boolean] Enabled
authType: # [string] Authentication method - Enter client secret directly, or se
clientSecret: # [string] Client secret - Office 365 Azure client secret

# -------------- if authType is manual ---------------


# -----------------------------------------------------

textSecret: # [string] Client secret (text secret) - Select or create a stored t

# -------------- if authType is secret ---------------


# -----------------------------------------------------

disabled: # [boolean] Disabled - Enable/disable this input
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

# -------------- if sendToRoutes is false ---------------

connections: # [array] Quick Connections - Direct connections to Destinations, o
  - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
    output: # [string] Destination - Select a Destination.

# -----------------------------------------------------

environment: # [string] Environment - Optionally, enable this config only on a s
pqEnabled: # [boolean] Enable Persistent Queue

# -------------- if pqEnabled is true ---------------

pq: # [object]
  mode: # [string] Mode - With Smart mode, PQ will write events to the filesyste
  maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
  commitFrequency: # [number] Commit frequency - The number of events to send do
  maxFileSize: # [string] Max file size - The maximum size to store in each queu
```

```
    maxSize: # [string] Max queue size - The maximum amount of disk space the queu
    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 - Add tags for filtering and grouping in S
office365_msg_trace_input: # [object]
  url: # [string] Report URL - URL to use when retrieving report data.
  interval: # [number] [required] Poll interval - How often (in minutes) to run th
  startDate: # [string] Date range start - Backward offset for the search range's
  endDate: # [string] Date range end - Backward offset for the search range's tail
  logLevel: # [string] Log level - Log Level (verbosity) for collection runtime bel
  timeout: # [number] Timeout (secs) - HTTP request inactivity timeout. Maximum is
  disableTimeFilter: # [boolean] Disable time filter - Disables time filtering of
  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 tha

  # ------------------------------------------------------


  # -------------- if authType is oauth ---------------

  clientSecret: # [string] Client secret - client_secret to pass in the OAuth requ
  tenantId: # [string] Tenant identifier - Directory ID (tenant identifier) in Mic
  clientId: # [string] Client ID - client_id to pass in the OAuth request paramete
  resource: # [string] Resource - Resource to pass in the OAuth request parameter.

  # ------------------------------------------------------


  # -------------- if authType is oauthSecret ---------------
```

```
textSecret: # [string] Client secret - Select or create a secret that references
tenantId: # [string] Tenant identifier - Directory ID (tenant identifier) in Mic
clientId: # [string] Client ID - client_id to pass in the OAuth request paramete
resource: # [string] Resource - Resource to pass in the OAuth request parameter.

# ------------------------------------------------------

keepAliveTime: # [number] Keep Alive Time (seconds) - How often workers should cl
maxMissedKeepAlives: # [number] Worker Timeout (periods) - The number of Keep Al
metadata: # [array] Fields - Fields to add to events from this input.
  - name: # [string] Name - Field name
    value: # [string] Value - JavaScript expression to compute field's value, en
type: # [string] Input Type
disabled: # [boolean] Disabled - Enable/disable this input
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

# -------------- if sendToRoutes is false ---------------

connections: # [array] Quick Connections - Direct connections to Destinations, o
  - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
    output: # [string] Destination - Select a Destination.

# ------------------------------------------------------

environment: # [string] Environment - Optionally, enable this config only on a s
pqEnabled: # [boolean] Enable Persistent Queue

# -------------- if pqEnabled is true ---------------

pq: # [object]
  mode: # [string] Mode - With Smart mode, PQ will write events to the filesyste
  maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
  commitFrequency: # [number] Commit frequency - The number of events to send do
  maxFileSize: # [string] Max file size - The maximum size to store in each queu
  maxSize: # [string] Max queue size - The maximum amount of disk space the queu
  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 - Add tags for filtering and grouping in S
```

```
eventhub_input: # [object]
  type: # [string] Input Type
  brokers: # [array of strings] Brokers - List of Event Hubs Kafka brokers to conn
  topics: # [array of strings] [required] Event Hub name - The name of the Event Hu
  groupId: # [string] Group ID - Specifies the consumer group this instance belongs
  fromBeginning: # [boolean] From beginning - Whether to start reading from earlies
  connectionTimeout: # [number] Connection timeout (ms) - Maximum time to wait for
  requestTimeout: # [number] Request timeout (ms) - Maximum time to wait for a succ
  sasl: # [object] Authentication - Authentication parameters to use when connecti
    disabled: # [boolean] Disabled - Enable authentication.

    # -------------- if disabled is false ---------------

    mechanism: # [string] SASL mechanism - SASL authentication mechanism to use. Pl
    username: # [string] Username - The username for authentication. For Event Hubs
    authType: # [string] Authentication method - Enter password directly, or selec

    # ---------------------------------------------------------

  tls: # [object] TLS settings (client side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    rejectUnauthorized: # [boolean] Validate server certs - For Event Hubs, this sh

    # ---------------------------------------------------------

  sessionTimeout: # [number] Session timeout (ms) -
    Timeout (a.k.a session.timeout.ms in Kafka domain) used to detect client failu
    If the client sends the broker no heartbeats before this 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] Rebalance timeout (ms) -
    Maximum allowed time (a.k.a rebalance.timeout.ms in Kafka domain) for each work
    If the timeout is exceeded, the coordinator broker will remove the worker from
    See details [here](https://github.com/Azure/azure-event-hubs-for-kafka/blob/mas
  heartbeatInterval: # [number] Heartbeat interval (ms) -
    Expected time (a.k.a heartbeat.interval.ms in Kafka domain) between heartbeats
    Value must be lower than sessionTimeout, and typically should not exceed 1/3 of
    See details [here](https://github.com/Azure/azure-event-hubs-for-kafka/blob/mas
  minimizeDuplicates: # [boolean] Minimize duplicates - Enable feature to minimize
  metadata: # [array] Fields - Fields to add to events from this input.
```

```
    - name: # [string] Name - Field name
      value: # [string] Value - JavaScript expression to compute field's value, en
  disabled: # [boolean] Disabled - Enable/disable this input
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Quick Connections - Direct connections to Destinations, o
    - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
      output: # [string] Destination - Select a Destination.

  # -------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a s
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyste
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
    commitFrequency: # [number] Commit frequency - The number of events to send do
    maxFileSize: # [string] Max file size - The maximum size to store in each queu
    maxSize: # [string] Max queue size - The maximum amount of disk space the queu
    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 - Add tags for filtering and grouping in S
exec_input: # [object]
  disabled: # [boolean] Disabled - Enable/disable this input
  command: # [string] Command - Command to execute; supports Bourne shell syntax
  retries: # [number] Max retries - Maximum number of retry attempts in the event
  scheduleType: # [string] Schedule type - Select a schedule type; either an inter

  # -------------- if scheduleType is interval ---------------

  interval: # [number] Interval - Interval between command executions in seconds.

  # -------------------------------------------------------
```

```
  # -------------- 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 b
  staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of tim
  metadata: # [array] Fields - Fields to add to events from this input.
    - name: # [string] Name - Field name
      value: # [string] Value - JavaScript expression to compute field's value, en
  type: # [string] Input Type
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

  # -------------- if sendToRoutes is false ----------------

  connections: # [array] Quick Connections - Direct connections to Destinations, o
    - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
      output: # [string] Destination - Select a Destination.

  # ---------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a s
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyste
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
    commitFrequency: # [number] Commit frequency - The number of events to send do
    maxFileSize: # [string] Max file size - The maximum size to store in each queu
    maxSize: # [string] Max queue size - The maximum amount of disk space the queu
    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 - Add tags for filtering and grouping in S
firehose_input: # [object]
  type: # [string] Input Type
```

```
disabled: # [boolean] Disabled - Enable/disable this input
host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses
port: # [number] [required] Port - Port to listen to.
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 name - The name of the predefined certi
  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) - Whether to require
  minVersion: # [string] Minimum TLS version - Minimum TLS version to accept from
  maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept from

  # ----------------------------------------------------

maxActiveReq: # [number] Max active requests - Maximum number of active requests
enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection i
captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add i
activityLogSampleRate: # [number] Activity log sample rate - How often request ac
requestTimeout: # [number] Request timeout (seconds) - How long to wait for an in
metadata: # [array] Fields - Fields to add to events from this input.
  - name: # [string] Name - Field name
    value: # [string] Value - JavaScript expression to compute field's value, enc
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] Quick Connections - Direct connections to Destinations, op
  - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
    output: # [string] Destination - Select a Destination.

# ----------------------------------------------------

environment: # [string] Environment - Optionally, enable this config only on a sp
pqEnabled: # [boolean] Enable Persistent Queue

# -------------- if pqEnabled is true ---------------
```

```
  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyste
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
    commitFrequency: # [number] Commit frequency - The number of events to send do
    maxFileSize: # [string] Max file size - The maximum size to store in each queu
    maxSize: # [string] Max queue size - The maximum amount of disk space the queu
    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 - Add tags for filtering and grouping in S
google_pubsub_input: # [object]
  type: # [string] Input Type
  topicName: # [string] Topic ID - ID of the topic to receive events from.
  subscriptionName: # [string] [required] Subscription ID - ID of the subscription
  createTopic: # [boolean] Create topic - If enabled, create topic if it does not
  createSubscription: # [boolean] Create subscription - If enabled, create subscri

  # -------------- if createSubscription is true ---------------

  orderedDelivery: # [boolean] Ordered delivery - If enabled, receive events in th

  # ------------------------------------------------------

  region: # [string] Region - Region to retrieve messages from. Select 'default' t
  googleAuthMethod: # [string] Authentication Method - Google authentication metho

  # -------------- if googleAuthMethod is manual ---------------

  serviceAccountCredentials: # [string] Service account credentials - Contents of

  # ------------------------------------------------------


  # -------------- if googleAuthMethod is secret ---------------

  secret: # [string] Service account credentials (text secret) - Select or create

  # ------------------------------------------------------

  maxBacklog: # [number] Max backlog - If Destination exerts backpressure, this se
```

```
  requestTimeout: # [number] Request timeout (ms) - Pull request timeout, in millis
  metadata: # [array] Fields - Fields to add to events from this input.
    - name: # [string] Name - Field name
      value: # [string] Value - JavaScript expression to compute field's value, en
  disabled: # [boolean] Disabled - Enable/disable this input
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Quick Connections - Direct connections to Destinations, o
    - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
      output: # [string] Destination - Select a Destination.

  # --------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a s
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyste
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
    commitFrequency: # [number] Commit frequency - The number of events to send do
    maxFileSize: # [string] Max file size - The maximum size to store in each queu
    maxSize: # [string] Max queue size - The maximum amount of disk space the queu
    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 - Add tags for filtering and grouping in S
cribl_input: # [object]
  type: # [string] Input Type
  metadata: # [array] Fields - Fields to add to events from this input.
    - name: # [string] Name - Field name
      value: # [string] Value - JavaScript expression to compute field's value, en
  disabled: # [boolean] Disabled - Enable/disable this input
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

  # -------------- if sendToRoutes is false ---------------
```

```
  connections: # [array] Quick Connections - Direct connections to Destinations, o
    - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
      output: # [string] Destination - Select a Destination.

  # ------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a s
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyste
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
    commitFrequency: # [number] Commit frequency - The number of events to send do
    maxFileSize: # [string] Max file size - The maximum size to store in each queu
    maxSize: # [string] Max queue size - The maximum amount of disk space the queu
    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 - Add tags for filtering and grouping in S
cribl_tcp_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled - Enable/disable this input
  host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresse
  port: # [number] [required] Port - Port to listen to.
  tls: # [object] TLS settings (server side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    certificateName: # [string] Certificate name - The name of the predefined cert
    privKeyPath: # [string] Private key path - Path on server containing the priva
    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 certific
    requestCert: # [boolean] Authenticate client (mutual auth) - Whether to requir
    minVersion: # [string] Minimum TLS version - Minimum TLS version to accept fro
    maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept fro
```

```
    # ------------------------------------------------------

  maxActiveCxn: # [number] Max Active Connections - Maximum number of active connec
  enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection i
  metadata: # [array] Fields - Fields to add to events from this input.
    - name: # [string] Name - Field name
      value: # [string] Value - JavaScript expression to compute field's value, enc
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly f

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Quick Connections - Direct connections to Destinations, op
    - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
      output: # [string] Destination - Select a Destination.

  # -------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a sp
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyster
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to hol
    commitFrequency: # [number] Commit frequency - The number of events to send dow
    maxFileSize: # [string] Max file size - The maximum size to store in each queue
    maxSize: # [string] Max queue size - The maximum amount of disk space the queue
    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 - Add tags for filtering and grouping in St
cribl_http_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled - Enable/disable this input
  host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses
  port: # [number] [required] Port - Port to listen to.
  authTokens: # [array of strings] Auth tokens - Shared secrets to be provided by a
  tls: # [object] TLS settings (server side)
    disabled: # [boolean] Disabled
```

```
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  certificateName: # [string] Certificate name - The name of the predefined certi
  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) - Whether to require
  minVersion: # [string] Minimum TLS version - Minimum TLS version to accept from
  maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept from

  # ---------------------------------------------------

maxActiveReq: # [number] Max active requests - Maximum number of active requests
enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection i
captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add i
activityLogSampleRate: # [number] Activity log sample rate - How often request ac
requestTimeout: # [number] Request timeout (seconds) - How long to wait for an in
metadata: # [array] Fields - Fields to add to events from this input.
  - name: # [string] Name - Field name
    value: # [string] Value - JavaScript expression to compute field's value, enc
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] Quick Connections - Direct connections to Destinations, op
  - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
    output: # [string] Destination - Select a Destination.

# ---------------------------------------------------------

environment: # [string] Environment - Optionally, enable this config only on a sp
pqEnabled: # [boolean] Enable Persistent Queue

# -------------- if pqEnabled is true ---------------

pq: # [object]
  mode: # [string] Mode - With Smart mode, PQ will write events to the filesyster
  maxBufferSize: # [number] Max buffer size - The maximum amount of events to hol
  commitFrequency: # [number] Commit frequency - The number of events to send dow
  maxFileSize: # [string] Max file size - The maximum size to store in each queue
```

```
    maxFileSize: # [string] Max file size - The maximum size to store in each queu
    maxSize: # [string] Max queue size - The maximum amount of disk space the queu
    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 - Add tags for filtering and grouping in S
tcpjson_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled - Enable/disable this input
  host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses
  port: # [number] [required] Port - Port to listen to.
  tls: # [object] TLS settings (server side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    certificateName: # [string] Certificate name - The name of the predefined certi
    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) - Whether to require
    minVersion: # [string] Minimum TLS version - Minimum TLS version to accept from
    maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept from

    # ------------------------------------------------------

  ipWhitelistRegex: # [string] IP Allowlist Regex - Regex matching IP addresses tha
  maxActiveCxn: # [number] Max Active Connections - Maximum number of active connec
  enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection i
  metadata: # [array] Fields - Fields to add to events from this input.
    - name: # [string] Name - Field name
      value: # [string] Value - JavaScript expression to compute field's value, enc
  authType: # [string] Authentication method - Enter a token directly, or provide a
  authToken: # [string] Auth token - Shared secret to be provided by any client (in

  # -------------- if authType is manual ---------------


  # ------------------------------------------------------

  textSecret: # [string] Auth token (text secret) - Select or create a stored text
```

```
  textSecret: # [string] Auth token (text secret) - Select or create a stored text

  # -------------- if authType is secret ---------------


  # -----------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Quick Connections - Direct connections to Destinations, o
    - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
      output: # [string] Destination - Select a Destination.

  # -----------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a s
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyste
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
    commitFrequency: # [number] Commit frequency - The number of events to send do
    maxFileSize: # [string] Max file size - The maximum size to store in each queu
    maxSize: # [string] Max queue size - The maximum amount of disk space the queu
    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 - Add tags for filtering and grouping in S
system_metrics_input: # [object]
  interval: # [number] Polling interval - Time, in seconds, between consecutive me
  host: # [object]
    mode: # [string]  - Select level of detail for host metrics

    # -------------- if mode is custom ---------------

    custom: # [object]
      system: # [object]
```

```
system: # [object]
  mode: # [string]  - Select the level of details for system metrics

  # -------------- if mode is custom ---------------

  processes: # [boolean] Process metrics - Generate metrics for the numbers

  # ------------------------------------------------------

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 coun

  # ------------------------------------------------------

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 sta

  # ------------------------------------------------------

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 inc
  perInterface: # [boolean] Per interface metrics - Generate separate metric
  detail: # [boolean] Detailed metrics - Generate full network metrics

  # ------------------------------------------------------

disk: # [object]
  mode: # [string]  - Select the level of details for disk metrics

  #               if mode is custom
```

```
    # -------------- if mode is custom ---------------

    devices: # [array of strings] Device filter - Block devices to include/exc
    mountpoints: # [array of strings] Mountpoint filter - Filesystem mountpoin
    fstypes: # [array of strings] Filesystem type filter - Filesystem types to
    perDevice: # [boolean] Per device metrics - Generate separate metrics for
    detail: # [boolean] Detailed metrics - Generate full disk metrics

    # -----------------------------------------------------


  # -------------------------------------------------------

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] Docker timeout - Timeout, in seconds, for the Docker

  # -------------------------------------------------------


  # -------------- if mode is all ---------------

  dockerSocket: # [array of strings] Docker socket - Full paths for Docker's UNI
  dockerTimeout: # [number] Docker timeout - Timeout, in seconds, for the Docker

  # -------------------------------------------------------


  # -------------- if mode is custom ---------------

  filters: # [array] Container Filters - Containers matching any of these will b
    - expr: # [string] Expression
  allContainers: # [boolean] All containers - Include stopped and paused contain
  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] Docker timeout - Timeout, in seconds, for the Docker

  # -------------------------------------------------------
```

```
metadata: # [array] Fields - Fields to add to events from this input.
  - name: # [string] Name - Field name
    value: # [string] Value - JavaScript expression to compute field's value, en
persistence: # [object] persistence
  enable: # [boolean] Enable disk persistence - Persist metrics on disk

  # -------------- if enable is true ---------------

  timeWindow: # [string] Bucket time span - Time span for each file bucket
  maxDataSize: # [string] Max data size - Maximum disk space allowed to be consu
  maxDataTime: # [string] Max data age - Maximum amount of time to retain data (
  compress: # [string] Compression - Select data compression format. Optional.
  destPath: # [string] Path location - Path to use to write metrics. Defaults to

  # ------------------------------------------------------

type: # [string] Input Type
disabled: # [boolean] Disabled - Enable/disable this input
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

# -------------- if sendToRoutes is false ---------------

connections: # [array] Quick Connections - Direct connections to Destinations, o
  - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
    output: # [string] Destination - Select a Destination.

# -------------------------------------------------------

environment: # [string] Environment - Optionally, enable this config only on a s
pqEnabled: # [boolean] Enable Persistent Queue

# -------------- if pqEnabled is true ---------------

pq: # [object]
  mode: # [string] Mode - With Smart mode, PQ will write events to the filesyste
  maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
  commitFrequency: # [number] Commit frequency - The number of events to send do
  maxFileSize: # [string] Max file size - The maximum size to store in each queu
  maxSize: # [string] Max queue size - The maximum amount of disk space the queu
  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 - Add tags for filtering and grouping in S
system_state_input: # [object]
  interval: # [number] Polling interval - Time, in seconds, between consecutive sta
  metadata: # [array] Fields - Fields to add to events from this input.
    - name: # [string] Name - Field name
      value: # [string] Value - JavaScript expression to compute field's value, en
  collectors: # [object]
    hostsfile: # [object]
      enable: # [boolean] Enabled
  persistence: # [object]
    enable: # [boolean] Enable disk persistence - Persist metrics on disk

    # -------------- if enable is true ---------------

    timeWindow: # [string] Bucket time span - Time span for each file bucket
    maxDataSize: # [string] Max data size - Maximum disk space allowed to be consur
    maxDataTime: # [string] Max data age - Maximum amount of time to retain data (
    compress: # [string] Compression - Select data compression format. Optional.
    destPath: # [string] Path location - Path to use to write metrics. Defaults to

    # ------------------------------------------------------

  type: # [string] Input Type
  disabled: # [boolean] Disabled - Enable/disable this input
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Quick Connections - Direct connections to Destinations, o
    - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
      output: # [string] Destination - Select a Destination.

  # ------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a s
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------
```

```
  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyste
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
    commitFrequency: # [number] Commit frequency - The number of events to send do
    maxFileSize: # [string] Max file size - The maximum size to store in each queu
    maxSize: # [string] Max queue size - The maximum amount of disk space the queu
    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 - Add tags for filtering and grouping in S
windows_metrics_input: # [object]
  interval: # [number] Polling interval - Time, in seconds, between consecutive me
  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 inf

        # ------------------------------------------------------------

      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 coun

        # ------------------------------------------------------------

      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 sta

      # -------------------------------------------------------

    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 inc
      perInterface: # [boolean] Per interface metrics - Generate separate metric
      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/e
      perVolume: # [boolean] Per volume metrics - Generate separate metrics for

      # -------------------------------------------------------


  # -----------------------------------------------------

metadata: # [array] Fields - Fields to add to events from this input.
  - name: # [string] Name - Field name
    value: # [string] Value - JavaScript expression to compute field's value, en
persistence: # [object] persistence
  enable: # [boolean] Enable disk persistence - Persist metrics on disk

  # -------------- if enable is true ---------------

  timeWindow: # [string] Bucket time span - Time span for each file bucket
  maxDataSize: # [string] Max data size - Maximum disk space allowed to be consu
  maxDataTime: # [string] Max data age - Maximum amount of time to retain data (
  compress: # [string] Compression - Select data compression format. Optional.
  destPath: # [string] Path location - Path to use to write metrics. Defaults to
```

```
    # -------------------------------------------------------

  type: # [string] Input Type
  disabled: # [boolean] Disabled - Enable/disable this input
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Quick Connections - Direct connections to Destinations, o
    - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
      output: # [string] Destination - Select a Destination.

  # -------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a s
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyste
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
    commitFrequency: # [number] Commit frequency - The number of events to send do
    maxFileSize: # [string] Max file size - The maximum size to store in each queu
    maxSize: # [string] Max queue size - The maximum amount of disk space the queu
    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 - Add tags for filtering and grouping in S
crowdstrike_input: # [object]
  type: # [string] Input Type
  queueName: # [string] Queue - The name, URL, or ARN of the SQS queue to read not
  fileFilter: # [string] Filename filter - Regex matching file names to download a
  awsAccountId: # [string] AWS Account ID - SQS queue owner's AWS account ID. Leav
  awsAuthenticationMethod: # [string] Authentication method - AWS authentication m

  # -------------- if awsAuthenticationMethod is manual ---------------

  awsApiKey: # [string] Access key - Access key
```

```
# ------------------------------------------------------


# -------------- if awsAuthenticationMethod is secret ---------------

awsSecret: # [string] Secret key pair - Select or create a stored secret that re

# ------------------------------------------------------

awsSecretKey: # [string] Secret key - Secret key
region: # [string] Region - AWS Region where the S3 bucket and SQS queue are loc
endpoint: # [string] Endpoint - S3 service endpoint. If empty, defaults to AWS'
signatureVersion: # [string] Signature version - Signature version to use for si
reuseConnections: # [boolean] Reuse connections - Whether to reuse connections b
rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to re
breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event b
staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of tim
maxMessages: # [number] Max Messages - The maximum number of messages SQS should
visibilityTimeout: # [number] Visibility timeout seconds - The duration (in seco
numReceivers: # [number] Num receivers - The Number of receiver processes to run
socketTimeout: # [number] Socket timeout - Socket inactivity timeout (in seconds
skipOnError: # [boolean] Skip file on error - Toggle to Yes to skip files that t
enableAssumeRole: # [boolean] Enable for S3 - Use Assume Role credentials to acc
assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the rol
assumeRoleExternalId: # [string] External ID - External ID to use when assuming
enableSQSAssumeRole: # [boolean] Enable for SQS - Use Assume Role credentials wh
preprocess: # [object]
  disabled: # [boolean] Disabled - Enable Custom Command
  command: # [string] Command - Command to feed the data through (via stdin) and
  args: # [array of strings] Arguments - Arguments
metadata: # [array] Fields - Fields to add to events from this input.
  - name: # [string] Name - Field name
    value: # [string] Value - JavaScript expression to compute field's value, en
pollTimeout: # [number] Poll timeout (secs) - The amount of time to wait for eve
disabled: # [boolean] Disabled - Enable/disable this input
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

# -------------- if sendToRoutes is false ---------------

connections: # [array] Quick Connections - Direct connections to Destinations, o
  - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
```

```
    output: # [string] Destination - Select a Destination.

  # ------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a sp
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyster
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
    commitFrequency: # [number] Commit frequency - The number of events to send dow
    maxFileSize: # [string] Max file size - The maximum size to store in each queu
    maxSize: # [string] Max queue size - The maximum amount of disk space the queu
    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 - Add tags for filtering and grouping in S
datadog_agent_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled - Enable/disable this input
  host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses
  port: # [number] [required] Port - Port to listen to.
  tls: # [object] TLS settings (server side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    certificateName: # [string] Certificate name - The name of the predefined certi
    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) - Whether to require
    minVersion: # [string] Minimum TLS version - Minimum TLS version to accept from
    maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept from

    # ------------------------------------------------------

  maxActiveReq: # [number] Max active requests - Maximum number of active requests
```

```
  enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
  captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add
  activityLogSampleRate: # [number] Activity log sample rate - How often request a
  requestTimeout: # [number] Request timeout (seconds) - How long to wait for an i
  extractMetrics: # [boolean] Extract metrics - Toggle to Yes to extract each inco
  metadata: # [array] Fields - Fields to add to events from this input.
    - name: # [string] Name - Field name
      value: # [string] Value - JavaScript expression to compute field's value, en
  proxyMode: # [object]
    enabled: # [boolean] Forward API key validation requests - Toggle to Yes to se
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Quick Connections - Direct connections to Destinations, o
    - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
      output: # [string] Destination - Select a Destination.

  # --------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a s
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyste
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
    commitFrequency: # [number] Commit frequency - The number of events to send do
    maxFileSize: # [string] Max file size - The maximum size to store in each queu
    maxSize: # [string] Max queue size - The maximum amount of disk space the queu
    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 - Add tags for filtering and grouping in S
datagen_input: # [object]
  samples: # [array] Datagen - List of datagens
    - sample: # [string] Data Generator File - Name of the datagen file
      eventsPerSec: # [number] Events Per Second Per Worker Node - Maximum no. of
  metadata: # [array] Fields - Fields to add to events from this input.
```

```
    - name: # [string] Name - Field name
      value: # [string] Value - JavaScript expression to compute field's value, en
  type: # [string] Input Type
  disabled: # [boolean] Disabled - Enable/disable this input
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Quick Connections - Direct connections to Destinations, o
    - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
      output: # [string] Destination - Select a Destination.

  # --------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a s
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyste
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
    commitFrequency: # [number] Commit frequency - The number of events to send do
    maxFileSize: # [string] Max file size - The maximum size to store in each queu
    maxSize: # [string] Max queue size - The maximum amount of disk space the queu
    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 - Add tags for filtering and grouping in S
http_raw_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled - Enable/disable this input
  host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresse
  port: # [number] [required] Port - Port to listen to.
  authTokens: # [array of strings] Auth tokens - Shared secrets to be provided by
  tls: # [object] TLS settings (server side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------
```

```
  certificateName: # [string] Certificate name - The name of the predefined certi
  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) - Whether to requir
  minVersion: # [string] Minimum TLS version - Minimum TLS version to accept fro
  maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept fro

  # ------------------------------------------------------

maxActiveReq: # [number] Max active requests - Maximum number of active requests
enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add
activityLogSampleRate: # [number] Activity log sample rate - How often request a
requestTimeout: # [number] Request timeout (seconds) - How long to wait for an i
breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event b
staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of tim
metadata: # [array] Fields - Fields to add to events from this input.
  - name: # [string] Name - Field name
    value: # [string] Value - JavaScript expression to compute field's value, en
allowedPaths: # [array of strings] Allowed URI paths - List of URI paths accepte
allowedMethods: # [array of strings] Allowed HTTP methods - List of HTTP methods
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

# -------------- if sendToRoutes is false ---------------

connections: # [array] Quick Connections - Direct connections to Destinations, o
  - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
    output: # [string] Destination - Select a Destination.

# -------------------------------------------------------

environment: # [string] Environment - Optionally, enable this config only on a s
pqEnabled: # [boolean] Enable Persistent Queue

# -------------- if pqEnabled is true ---------------

pq: # [object]
  mode: # [string] Mode - With Smart mode, PQ will write events to the filesyste
  maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
  commitFrequency: # [number] Commit frequency - The number of events to send do
```

```
    maxFileSize: # [string] Max file size - The maximum size to store in each queu
    maxSize: # [string] Max queue size - The maximum amount of disk space the queu
    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 - Add tags for filtering and grouping in S
kinesis_input: # [object]
  type: # [string] Input Type
  streamName: # [string] Stream name - Kinesis stream name to read data from.
  serviceInterval: # [number] Service Period - Time interval in minutes between co
  shardExpr: # [string] Shard selection expression - A JS expression to be called
  shardIteratorType: # [string] Shard iterator start - Location at which to start
  payloadFormat: # [string] Record data format - Format of data inside the Kinesis
  awsAuthenticationMethod: # [string] Authentication method - AWS authentication m

  # -------------- if awsAuthenticationMethod is manual ---------------

  awsApiKey: # [string] Access key - Access key

  # ------------------------------------------------------


  # -------------- if awsAuthenticationMethod is secret ---------------

  awsSecret: # [string] Secret key pair - Select or create a stored secret that re

  # ------------------------------------------------------


  awsSecretKey: # [string] Secret key - Secret key
  region: # [string] [required] Region - Region where the Kinesis stream is locate
  endpoint: # [string] Endpoint - Kinesis stream service endpoint. If empty, defau
  signatureVersion: # [string] Signature version - Signature version to use for si
  reuseConnections: # [boolean] Reuse connections - Whether to reuse connections b
  rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to re
  enableAssumeRole: # [boolean] Enable for Kinesis stream - Use Assume Role creden
  assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the rol
  assumeRoleExternalId: # [string] External ID - External ID to use when assuming
  verifyKPLCheckSums: # [boolean] Verify KPL checksums - Verify Kinesis Producer L
  avoidDuplicates: # [boolean] Avoid duplicate records - Yes means: when resuming
  metadata: # [array] Fields - Fields to add to events from this input.
    - name: # [string] Name - Field name
```

```
    value: # [string] Value - JavaScript expression to compute field's value, en
  disabled: # [boolean] Disabled - Enable/disable this input
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Quick Connections - Direct connections to Destinations, o
    - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
      output: # [string] Destination - Select a Destination.

  # --------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a s
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyste
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
    commitFrequency: # [number] Commit frequency - The number of events to send do
    maxFileSize: # [string] Max file size - The maximum size to store in each queu
    maxSize: # [string] Max queue size - The maximum amount of disk space the queu
    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 - Add tags for filtering and grouping in S
logstream_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled - Enable/disable this input
  host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresse
  port: # [number] [required] Port - Port to listen to.
  tls: # [object] TLS settings (server side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    certificateName: # [string] Certificate name - The name of the predefined cert
    privKeyPath: # [string] Private key path - Path on server containing the priva
    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) - Whether to requir
  minVersion: # [string] Minimum TLS version - Minimum TLS version to accept fro
  maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept fro

  # ------------------------------------------------------

ipWhitelistRegex: # [string] IP Allowlist Regex - Regex matching IP addresses tha
maxActiveCxn: # [number] Max Active Connections - Maximum number of active conne
enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
metadata: # [array] Fields - Fields to add to events from this input.
  - name: # [string] Name - Field name
    value: # [string] Value - JavaScript expression to compute field's value, en
authType: # [string] Authentication method - Enter a token directly, or provide a
authToken: # [string] Auth token - Shared secret to be provided by any client (i

# -------------- if authType is manual ---------------


# -------------------------------------------------------

textSecret: # [string] Auth token (text secret) - Select or create a stored text

# -------------- if authType is secret ---------------


# -------------------------------------------------------

pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

# -------------- if sendToRoutes is false ---------------

connections: # [array] Quick Connections - Direct connections to Destinations, o
  - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
    output: # [string] Destination - Select a Destination.

# -------------------------------------------------------

environment: # [string] Environment - Optionally, enable this config only on a s
pqEnabled: # [boolean] Enable Persistent Queue
```

```
  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyste
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
    commitFrequency: # [number] Commit frequency - The number of events to send do
    maxFileSize: # [string] Max file size - The maximum size to store in each queu
    maxSize: # [string] Max queue size - The maximum amount of disk space the queu
    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 - Add tags for filtering and grouping in S
criblmetrics_input: # [object]
  type: # [string] Input Type
  prefix: # [string] Metric Name Prefix - A prefix that is applied to the metrics
  fullFidelity: # [boolean] Full Fidelity - Include granular metrics.  Disabling t
  metadata: # [array] Fields - Fields to add to events from this input.
    - name: # [string] Name - Field name
      value: # [string] Value - JavaScript expression to compute field's value, en
  disabled: # [boolean] Disabled - Enable/disable this input
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Quick Connections - Direct connections to Destinations, o
    - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
      output: # [string] Destination - Select a Destination.

  # ------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a s
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyste
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
    commitFrequency: # [number] Commit frequency - The number of events to send do
    maxFileSize: # [string] Max file size - The maximum size to store in each queu
```

```
    maxSize: # [string] Max queue size - The maximum amount of disk space the queu
    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 - Add tags for filtering and grouping in S
metrics_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled - Enable/disable this input
  host: # [string] Address - Address to bind on. For IPv4 (all addresses), use the
  udpPort: # [number] UDP Port - Enter UDP port number to listen on. Not required
  tcpPort: # [number] TCP Port - Enter TCP port number to listen on. Not required
  maxBufferSize: # [number] Max Buffer Size (events) - Maximum number of events to
  ipWhitelistRegex: # [string] IP Allowlist Regex - Regex matching IP addresses tha
  enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
  tls: # [object] TLS settings (server side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    certificateName: # [string] Certificate name - The name of the predefined cert
    privKeyPath: # [string] Private key path - Path on server containing the priva
    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) - Whether to requir
    minVersion: # [string] Minimum TLS version - Minimum TLS version to accept fro
    maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept fro

    # ------------------------------------------------------

  metadata: # [array] Fields - Fields to add to events from this input.
    - name: # [string] Name - Field name
      value: # [string] Value - JavaScript expression to compute field's value, en
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Quick Connections - Direct connections to Destinations, o
    - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
      output: # [string] Destination - Select a Destination.
```

```
  # ------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a sp
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyster
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
    commitFrequency: # [number] Commit frequency - The number of events to send dow
    maxFileSize: # [string] Max file size - The maximum size to store in each queue
    maxSize: # [string] Max queue size - The maximum amount of disk space the queue
    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 - Add tags for filtering and grouping in S
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 - Access key

  # ------------------------------------------------------



  # -------------- if awsAuthenticationMethod is secret ---------------

  awsSecret: # [string] Secret key pair - Select or create a stored secret that re

  # ------------------------------------------------------

  awsSecretKey: # [string] Secret key - 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 AWS' R
```

```
signatureVersion: # [string] Signature version - Signature version to use for sig
reuseConnections: # [boolean] Reuse connections - Whether to reuse connections b
rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to re
breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event b
staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of tim
maxMessages: # [number] Max Messages - The maximum number of messages SQS should
visibilityTimeout: # [number] Visibility timeout seconds - The duration (in seco
numReceivers: # [number] Num receivers - The Number of receiver processes to run
socketTimeout: # [number] Socket timeout - Socket inactivity timeout (in seconds
skipOnError: # [boolean] Skip file on error - Toggle to Yes to skip files that t
enableAssumeRole: # [boolean] Enable for S3 - Use Assume Role credentials to acc
assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the rol
assumeRoleExternalId: # [string] External ID - External ID to use when assuming
enableSQSAssumeRole: # [boolean] Enable for SQS - Use Assume Role credentials wh
preprocess: # [object]
  disabled: # [boolean] Disabled - Enable Custom Command
  command: # [string] Command - Command to feed the data through (via stdin) and
  args: # [array of strings] Arguments - Arguments
metadata: # [array] Fields - Fields to add to events from this input.
  - name: # [string] Name - Field name
    value: # [string] Value - JavaScript expression to compute field's value, en
parquetChunkSizeMB: # [number] Max Parquet chunk size (MB) - Maximum file size f
parquetChunkDownloadTimeout: # [number] Parquet chunk download timeout (seconds)
pollTimeout: # [number] Poll timeout (secs) - The amount of time to wait for eve
disabled: # [boolean] Disabled - Enable/disable this input
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

# -------------- if sendToRoutes is false ---------------

connections: # [array] Quick Connections - Direct connections to Destinations, o
  - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
    output: # [string] Destination - Select a Destination.

# -------------------------------------------------------

environment: # [string] Environment - Optionally, enable this config only on a s
pqEnabled: # [boolean] Enable Persistent Queue

# -------------- if pqEnabled is true ---------------

pq: # [object]
  mode: # [string] Mode - With Smart mode, PQ will write events to the filesyste
```

```
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyste
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
    commitFrequency: # [number] Commit frequency - The number of events to send do
    maxFileSize: # [string] Max file size - The maximum size to store in each queu
    maxSize: # [string] Max queue size - The maximum amount of disk space the queu
    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 - Add tags for filtering and grouping in S
snmp_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled - Enable/disable this input
  host: # [string] Address - Address to bind on. For IPv4 (all addresses), use the
  port: # [number] [required] UDP Port - UDP port to receive SNMP traps on. Defaul
  maxBufferSize: # [number] Max Buffer Size (events) - Maximum number of events to
  ipWhitelistRegex: # [string] IP Allowlist Regex - Regex matching IP addresses tha
  metadata: # [array] Fields - Fields to add to events from this input.
    - name: # [string] Name - Field name
      value: # [string] Value - JavaScript expression to compute field's value, en
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Quick Connections - Direct connections to Destinations, o
    - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
      output: # [string] Destination - Select a Destination.

  # ------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a s
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyste
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
    commitFrequency: # [number] Commit frequency - The number of events to send do
    maxFileSize: # [string] Max file size - The maximum size to store in each queu
    maxSize: # [string] Max queue size - The maximum amount of disk space the queu
    path: # [string] Queue file path - The location for the persistent queue files
```

```
    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 - Add tags for filtering and grouping in S
open_telemetry_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled - Enable/disable this input
  host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses
  port: # [number] [required] Port - Port to listen to.
  tls: # [object] TLS settings (server side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    certificateName: # [string] Certificate name - The name of the predefined certi
    privKeyPath: # [string] Private key path - Path on server containing the priva
    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) - Whether to requir

    # ------------------------------------------------------

  maxActiveReq: # [number] Max active requests - Maximum number of active requests
  enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection i
  captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add i
  activityLogSampleRate: # [number] Activity log sample rate - How often request ac
  requestTimeout: # [number] Request timeout (seconds) - How long to wait for an in
  extractSpans: # [boolean] Extract spans - Toggle to Yes to extract each incoming
  extractMetrics: # [boolean] Extract metrics - Toggle to Yes to extract each incom
  maxActiveCxn: # [number] Max active connections - Maximum number of active connec
  authType: # [string] Authentication type - OpenTelemetry authentication type

  # -------------- if authType is basic ---------------

  username: # [string] Username - Username for Basic authentication
  password: # [string] Password - Password for Basic authentication

  # ------------------------------------------------------


  # -------------- 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 tha

# ------------------------------------------------------

# -------------- if authType is textSecret ---------------

textSecret: # [string] Token (text secret) - Select or create a stored text secr

# ------------------------------------------------------

metadata: # [array] Fields - Fields to add to events from this input.
  - name: # [string] Name - Field name
    value: # [string] Value - JavaScript expression to compute field's value, en
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

# -------------- if sendToRoutes is false ---------------

connections: # [array] Quick Connections - Direct connections to Destinations, o
  - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
    output: # [string] Destination - Select a Destination.

# ------------------------------------------------------

environment: # [string] Environment - Optionally, enable this config only on a s
pqEnabled: # [boolean] Enable Persistent Queue

# -------------- if pqEnabled is true ---------------

pq: # [object]
  mode: # [string] Mode - With Smart mode, PQ will write events to the filesyste
  maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
  commitFrequency: # [number] Commit frequency - The number of events to send do
  maxFileSize: # [string] Max file size - The maximum size to store in each queu
  maxSize: # [string] Max queue size - The maximum amount of disk space the queu
```

```
    maxSize: # [string] Max queue size - The maximum amount of disk space the queu
    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 - Add tags for filtering and grouping in S
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). I

  # -------------- if queueType is standard ---------------

  numReceivers: # [number] Num receivers - The Number of receiver processes to run

  # ------------------------------------------------------

  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 - Access key

  # ------------------------------------------------------


  # -------------- if awsAuthenticationMethod is secret ---------------

  awsSecret: # [string] Secret key pair - Select or create a stored secret that ref

  # ------------------------------------------------------

  awsSecretKey: # [string] Secret key - Secret key
  region: # [string] Region - AWS Region where the SQS queue is located. Required,
  endpoint: # [string] Endpoint - SQS service endpoint. If empty, defaults to AWS'
  signatureVersion: # [string] Signature version - Signature version to use for sig
  reuseConnections: # [boolean] Reuse connections - Whether to reuse connections be
  rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to rej
  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
```

```
  assumeRoleExternalId: # [string] External ID - External ID to use when assuming
  maxMessages: # [number] Max Messages - The maximum number of messages SQS should
  visibilityTimeout: # [number] Visibility Timeout Seconds - The duration (in seco
  metadata: # [array] Fields - Fields to add to events from this input.
    - name: # [string] Name - Field name
      value: # [string] Value - JavaScript expression to compute field's value, en
  pollTimeout: # [number] Poll timeout (secs) - The amount of time to wait for eve
  disabled: # [boolean] Disabled - Enable/disable this input
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Quick Connections - Direct connections to Destinations, o
    - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
      output: # [string] Destination - Select a Destination.

  # --------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a s
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyste
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
    commitFrequency: # [number] Commit frequency - The number of events to send do
    maxFileSize: # [string] Max file size - The maximum size to store in each queu
    maxSize: # [string] Max queue size - The maximum amount of disk space the queu
    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 - Add tags for filtering and grouping in S
syslog_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled - Enable/disable this input
  host: # [string] Address - Address to bind on. For IPv4 (all addresses), use the
  udpPort: # [number] UDP port - Enter UDP port number to listen on. Not required
  tcpPort: # [number] TCP port - Enter TCP port number to listen on. Not required
  maxBufferSize: # [number] Max buffer size (events) - Maximum number of events to
  ipWhitelistRegex: # [string] IP allowlist regex - Regex matching IP addresses tha
```

```
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 - Whether to treat UDP packet
enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection i
keepFieldsList: # [array of strings] Fields to keep - Wildcard list of fields to
octetCounting: # [boolean] Octet count framing - Enable if incoming messages use
tls: # [object] TLS settings (server side)
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  certificateName: # [string] Certificate name - The name of the predefined certi
  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) - Whether to require
  minVersion: # [string] Minimum TLS version - Minimum TLS version to accept from
  maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept from

  # ----------------------------------------------------------

metadata: # [array] Fields - Fields to add to events from this input.
  - name: # [string] Name - Field name
    value: # [string] Value - JavaScript expression to compute field's value, enc
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly f

# -------------- if sendToRoutes is false ---------------

connections: # [array] Quick Connections - Direct connections to Destinations, op
  - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
    output: # [string] Destination - Select a Destination.

# ----------------------------------------------------------

environment: # [string] Environment - Optionally, enable this config only on a sp
pqEnabled: # [boolean] Enable Persistent Queue

# -------------- if pqEnabled is true ---------------

pq: # [object]
  mode: # [string] Mode - With Smart mode, PQ will write events to the filesyster
```

```
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
    commitFrequency: # [number] Commit frequency - The number of events to send dow
    maxFileSize: # [string] Max file size - The maximum size to store in each queu
    maxSize: # [string] Max queue size - The maximum amount of disk space the queu
    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 - Add tags for filtering and grouping in S
file_input: # [object]
  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

  # ------------------------------------------------------

  interval: # [number] Polling interval - Time, in seconds, between scanning for fi
  filenames: # [array of strings] Filename allowlist - The full path of discovered
  metadata: # [array] Fields - Fields to add to events from this input.
    - name: # [string] Name - Field name
      value: # [string] Value - JavaScript expression to compute field's value, en
  breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event br
  staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of time
  type: # [string] Input Type
  disabled: # [boolean] Disabled - Enable/disable this input
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Quick Connections - Direct connections to Destinations, op
    - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
      output: # [string] Destination - Select a Destination.

  # ------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a sp
  pqEnabled: # [boolean] Enable Persistent Queue
```

```
  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyster
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
    commitFrequency: # [number] Commit frequency - The number of events to send dow
    maxFileSize: # [string] Max file size - The maximum size to store in each queu
    maxSize: # [string] Max queue size - The maximum amount of disk space the queu
    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 - Add tags for filtering and grouping in S
tcp_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled - Enable/disable this input
  host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses
  port: # [number] [required] Port - Port to listen to.
  tls: # [object] TLS settings (server side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    certificateName: # [string] Certificate name - The name of the predefined cert
    privKeyPath: # [string] Private key path - Path on server containing the priva
    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) - Whether to requir
    minVersion: # [string] Minimum TLS version - Minimum TLS version to accept fro
    maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept fro

    # ------------------------------------------------------

  ipWhitelistRegex: # [string] IP Allowlist Regex - Regex matching IP addresses tha
  maxActiveCxn: # [number] Max Active Connections - Maximum number of active conne
  enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
  metadata: # [array] Fields - Fields to add to events from this input.
    - name: # [string] Name - Field name
      value: # [string] Value - JavaScript expression to compute field's value, en
  breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event b
  staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of tim
```

```
  enableHeader: # [boolean] Enable Header - If enabled, client will pass the heade

  # -------------- if enableHeader is true ---------------

  authType: # [string] Authentication method - Enter a token directly, or provide a

  # ------------------------------------------------------

  preprocess: # [object]
    disabled: # [boolean] Disabled - Enable Custom Command
    command: # [string] Command - Command to feed the data through (via stdin) and
    args: # [array of strings] Arguments - Arguments
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Quick Connections - Direct connections to Destinations, op
    - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
      output: # [string] Destination - Select a Destination.

  # ------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a sp
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyster
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
    commitFrequency: # [number] Commit frequency - The number of events to send dow
    maxFileSize: # [string] Max file size - The maximum size to store in each queue
    maxSize: # [string] Max queue size - The maximum amount of disk space the queue
    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 - Add tags for filtering and grouping in S
appscope_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled - Enable/disable this input
```

```
ipWhitelistRegex: # [string] IP Allowlist Regex - Regex matching IP addresses tha
maxActiveCxn: # [number] Max Active Connections - Maximum number of active connec
enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection i
metadata: # [array] Fields - Fields to add to events from this input.
  - name: # [string] Name - 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] Event Breaker buffer timeout - The amount of time
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] Port - Port to listen to.
tls: # [object] TLS settings (server side)
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  certificateName: # [string] Certificate name - The name of the predefined certi
  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) - Whether to require
  minVersion: # [string] Minimum TLS version - Minimum TLS version to accept from
  maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept from

  # ---------------------------------------------------------


# ---------------------------------------------------------


# -------------- 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

# ---------------------------------------------------------

authType: # [string] Authentication method - Enter a token directly, or provide a
authToken: # [string] Auth token - Shared secret to be provided by any client (in
```

```
  # -------------- if authType is manual ---------------


  # ------------------------------------------------------

  textSecret: # [string] Auth token (text secret) - Select or create a stored text

  # -------------- if authType is secret ---------------


  # ------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Quick Connections - Direct connections to Destinations, o
    - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
      output: # [string] Destination - Select a Destination.

  # ------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a s
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyste
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
    commitFrequency: # [number] Commit frequency - The number of events to send do
    maxFileSize: # [string] Max file size - The maximum size to store in each queu
    maxSize: # [string] Max queue size - The maximum amount of disk space the queu
    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 - Add tags for filtering and grouping in S
wef_input: # [object]
  type: # [string] Input Type
```

```
disabled: # [boolean] Disabled - Enable/disable this input
host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses
port: # [number] [required] Port - Port to listen to.
tls: # [object] [required] mTLS settings
  disabled: # [boolean] Disabled - Enable TLS
  rejectUnauthorized: # [boolean] Validate client certs - Required for WEF certi
  requestCert: # [boolean] Authenticate client - Required for WEF certificate au
  certificateName: # [string] Certificate name - Name of the predefined certifica
  privKeyPath: # [string] Private key path - Path on server containing the priva
  passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
  certPath: # [string] [required] Certificate path - Path on server containing c
  caPath: # [string] [required] CA certificate path - Path on server containing
  commonNameRegex: # [string] Common name - Regex matching allowable common names
  minVersion: # [string] Minimum TLS version - Minimum TLS version to accept fro
  maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept fro
maxActiveReq: # [number] Max active requests - Maximum number of active requests
enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add
caFingerprint: # [string] CA fingerprint override - SHA1 fingerprint expected by
allowMachineIdMismatch: # [boolean] Allow MachineID mismatch - Allow events to b
subscriptions: # [array] [required] Subscriptions - Subscriptions to events on f
  - subscriptionName: # [string] Name - Friendly name for this subscription.
    version: # [string] Version - Version UUID for this subscription. If any subs
    contentFormat: # [string] Format - Content format in which the endpoint shou
    heartbeatInterval: # [number] Heartbeat - Max time (in seconds) between endp
    batchTimeout: # [number] Batch timeout - Interval (in seconds) over which th
    readExistingEvents: # [boolean] Read existing events - Set to Yes if a newly
    sendBookmarks: # [boolean] Use bookmarks - If toggled to Yes, Cribl Edge wil
    compress: # [boolean] Compression - If toggled to Yes, Stream will receive c
    targets: # [array of strings] Targets - Enter the DNS names of the endpoints
    querySelector: # [string] Query builder mode - Select the query builder mode
pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

# -------------- if sendToRoutes is false ---------------

connections: # [array] Quick Connections - Direct connections to Destinations, o
  - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
    output: # [string] Destination - Select a Destination.

# -------------------------------------------------------

environment: # [string] Environment - Optionally, enable this config only on a s
```

```
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyster
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
    commitFrequency: # [number] Commit frequency - The number of events to send do
    maxFileSize: # [string] Max file size - The maximum size to store in each queu
    maxSize: # [string] Max queue size - The maximum amount of disk space the queu
    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 - Add tags for filtering and grouping in S
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
  interval: # [number] Polling Interval - Time, in seconds, between checking for n
  batchSize: # [number] Batch Size - Maximum number of event records to read in on
  metadata: # [array] Fields - Fields to add to events from this input.
    - name: # [string] Name - Field name
      value: # [string] Value - JavaScript expression to compute field's value, en
  disabled: # [boolean] Disabled - Enable/disable this input
  pipeline: # [string] Pipeline - Pipeline to process data from this Source before
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Quick Connections - Direct connections to Destinations, o
    - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
      output: # [string] Destination - Select a Destination.

  # ----------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a s
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
```

```
  mode: # [string] Mode - With Smart mode, PQ will write events to the filesyster
  maxBufferSize: # [number] Max buffer size - The maximum amount of events to ho
  commitFrequency: # [number] Commit frequency - The number of events to send do
  maxFileSize: # [string] Max file size - The maximum size to store in each queu
  maxSize: # [string] Max queue size - The maximum amount of disk space the queu
  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 - Add tags for filtering and grouping in S
```

15.4.6. Instance.Yml

Instance configuration is located under `$CRIBL_HOME/local/_system/instance.yml` (`C:\ProgramData\Cribl\local\_system.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: # [string] One of: master | worker | single | edge | managed-edge.
  master: # [object]
    disableSNIRouting: # [boolean] Whether SNI routing is enabled. See (Environment V
    host: # [string] Instance host address.
    port: # [number] Instance post number.
    tls: # [object]
      disabled: # [boolean]
      certificateName: # [string] TLS certificate name.
      rejectUnauthorized: # [boolean] False if ignoring untrusted certificates.
      requestCert: # [boolean] Whether certificates are required and validated.
      privKeyPath: # [string] Path to private key file. Can reference $ENV_VARS.
      certPath: # [string] Path to certificate file. Can reference $ENV_VARS.
      caPath: # [string] Path to CA certificate file. Can reference $ENV_VARS.
      commonNameRegex: # [any] Allowed common names in peer certificates' subject (re
      # Available values: 'TLSv1.3' | 'TLSv1.2' | 'TLSv1.1' | 'TLSv1'
      minVersion: # [string] Minimum TLS version.
      maxVersion: # [string] Maximum TLS version.
      servername: # [string] Server name.
    resiliency: # [string] Preferred failover mode, one of: none | failover.
    ipWhitelistRegex: # [string] IP addresses allowed to send data (regex).
    maxActiveCxn: # [number] Max number of active connections allowed per Worker Proc
    authToken: # [string] Token used for communication between Leader and managed nod
    compression: # [string] Codec used to compress persisted data. One of: none | gzi
    connectionTimeout: # [number] Maximum time to wait for a connection to complete s
    writeTimeout: # [number] Time (ms) to wait for a write to complete before assumin
    configBundles: # [object]
      remoteUrl: # [string] Bucket to use for remote bundle storage, in s3://${bucket
  group: # [string] Group the managed node belongs to.
  envRegex: # [string] Regex used to filter env variable names that can be sent by ma
  tags: # [array of strings] Tags associated with managed node, can be used by Leade
```

15.4.7. 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.

```
$CRIBL_HOME/local/cribl/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]
      outputName: # [string] Auto-populate from - The name of the predefined Destina
      authType: # [string] Authentication method - Enter authentication data directl

      # -------------- if authType is manual ---------------

      connectionString: # [string] Connection string - Enter your Azure storage acco

      # -----------------------------------------------------


      # -------------- if authType is secret ---------------

      textSecret: # [string] Connection string (text secret) - Text secret

      # -----------------------------------------------------

      containerName: # [string] Container name - Name of the container to collect fr
      path: # [string] Path - The directory from which to collect data. Templating i
      extractors: # [array] Path extractors - Allows using template tokens as contex
        - key: # [string] Token - A token from the template path, e.g.: epoch
          expression: # [string] Extractor expression - JS expression that receives
      recurse: # [boolean] Recursive - Whether to recurse through subdirectories.
      maxBatchSize: # [number] Max Batch Size (objects) - Maximum number of metadata

    # -----------------------------------------------------


    # -------------- if type is filesystem ---------------

    conf: # [object] [required]
      outputName: # [string] Auto-populate from - Select a predefined configuration
      path: # [string] Directory - The directory from which to collect data. Templat
      extractors: # [array] Path extractors - Allows using template tokens as contex
        - key: # [string] Token - A token from the template directory, e.g.: epoch
```

```
      expression: # [string] Extractor expression - JS expression that receives
  recurse: # [boolean] Recursive - Whether to recurse through subdirectories.
  maxBatchSize: # [number] Max Batch Size (files) - Maximum number of metadata f

# ------------------------------------------------------


# -------------- if type is google_cloud_storage ---------------

conf: # [object] [required]
  outputName: # [string] Auto-populate from - The name of the predefined Destina
  bucket: # [string] Bucket name - Name of the bucket to collect from. This valu
  path: # [string] Path - The directory from which to collect data. Templating i
  extractors: # [array] Path extractors - Allows using template tokens as contex
    - key: # [string] Token - A token from the template path, e.g.: epoch
      expression: # [string] Extractor expression - JS expression that receives
  endpoint: # [string] Endpoint - Google Cloud Storage service endpoint. If empt
  disableTimeFilter: # [boolean] Disable time filter - Used to disable collector
  recurse: # [boolean] Recursive - Whether to recurse through subdirectories.
  maxBatchSize: # [number] Max Batch Size (objects) - Maximum number of metadata
  authType: # [string] Authentication method - Enter account credentials manuall

  # -------------- if authType is manual ---------------

  serviceAccountCredentials: # [string] Service account credentials - Contents o

  # ------------------------------------------------------


  # -------------- if authType is secret ---------------

  textSecret: # [string] Service account credentials (text secret) - Select or c

  # ------------------------------------------------------


# ------------------------------------------------------


# -------------- 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 sen

# ------------------------------------------------------

collectRequestHeaders: # [array] Health check headers - Optional health check
  - name: # [string] Name - Header Name
    value: # [string] Value - JavaScript expression to compute the header valu
authenticateCollect: # [boolean] Authenticate health check - Enable to make au
authentication: # [string] [required] Authentication - Authentication method f

# -------------- 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 s

# ------------------------------------------------------


# -------------- if authentication is login ---------------

loginUrl: # [string] Login URL - URL to use for login API call. This call is e
username: # [string] Username - Login username
password: # [string] Password - Login password
loginBody: # [string] POST Body - Template for POST body to send with login re
tokenRespAttribute: # [string] Token Attribute - Path to token attribute in lo
authHeaderExpr: # [string] Authorize Expression - JavaScript expression to com
authRequestHeaders: # [array] Authentication Headers - Optional authentication
  - name: # [string] Name - Header name.
```

```
    value: # [string] Value - JavaScript expression to compute the header valu

# ------------------------------------------------------


# -------------- if authentication is loginSecret ---------------

loginUrl: # [string] Login URL - URL to use for login API call, this call is e
credentialsSecret: # [string] Credentials secret - Select or create a stored s
loginBody: # [string] POST Body - Template for POST body to send with login re
tokenRespAttribute: # [string] Token Attribute - Path to token attribute in lo
authHeaderExpr: # [string] Authorize Expression - JavaScript expression to com
authRequestHeaders: # [array] Authentication Headers - Optional authentication
  - name: # [string] Name - Header name.
    value: # [string] Value - JavaScript expression to compute the header valu

# ------------------------------------------------------


# -------------- if authentication is oauth ---------------

loginUrl: # [string] Login URL - URL to use for the OAuth API call. This call
tokenRespAttribute: # [string] Token attribute - Path to token attribute in lo
authHeaderExpr: # [string] Authorize expression - JavaScript expression to com
clientSecretParamName: # [string] Client secret parameter - Parameter name tha
clientSecretParamValue: # [string] Client secret value - Secret value to add t
authRequestParams: # [array] Extra authentication parameters - OAuth request p
  - 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 va

# ------------------------------------------------------


# -------------- if authentication is oauthSecret ---------------

loginUrl: # [string] Login URL - URL to use for the OAuth API call. This call
tokenRespAttribute: # [string] Token attribute - Path to token attribute in lo
authHeaderExpr: # [string] Authorize expression - JavaScript expression to com
clientSecretParamName: # [string] Client secret parameter - Parameter name tha
textSecret: # [string] Client secret value (text secret) - Select or create a
```

```
  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 va

  # ------------------------------------------------------

  timeout: # [number] Request Timeout (secs) - HTTP request inactivity timeout, d
  defaultBreakers: # [string] Hidden Default Breakers
  safeHeaders: # [array of strings] Safe headers - List of headers that are safe

# ------------------------------------------------------


# -------------- if type is office365_mgmt ---------------

conf: # [object] [required]
  plan_type: # [string] Subscription plan - Office 365 subscription plan for you
  tenant_id: # [string] [required] Tenant identifier - Directory ID (tenant iden
  app_id: # [string] [required] Application identifier - Identifier of the regis
  client_secret: # [string] [required] Client secret - Application key of the reg
  publisher_identifier: # [string] Publisher identifier - Optional PublisherIden
  content_type: # [string] [required] Content type - The type of content to retr

# ------------------------------------------------------


# -------------- if type is office365_service ---------------

conf: # [object] [required]
  tenant_id: # [string] Tenant identifier - Directory ID (tenant identifier) in M
  app_id: # [string] [required] Application identifier - Identifier of the regis
  client_secret: # [string] [required] Client secret - Application key of the reg
  content_type: # [string] [required] Content type - The type of content to retr

# ------------------------------------------------------


# -------------- if type is prometheus ---------------

conf: # [object] [required]
```

```
  dimensionList: # [array] Extra Dimensions - Other dimensions to include in evei
  username: # [string] Username - Optional username for Basic authentication
  password: # [string] Password - Optional password for Basic authentication
  discoveryType: # [string] Discovery Type - Target discovery mechanism, use sta

  # -------------- if discoveryType is static ---------------

  targetList: # [array] Targets - List of Prometheus targets to pull metrics fro

  # ----------------------------------------------------------


  # -------------- 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 r
  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
  scrapeProtocol: # [string] Metrics Protocol - Protocol to use when collecting r
  scrapePort: # [number] Metrics Port - The port number in the metrics URL for di
  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://
      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]
  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 parameter
    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] Collect URL - Expression to derive URL to use for the co
  collectMethod: # [string] [required] Collect method - Collect HTTP method.

  # -------------- if collectMethod is get ---------------

  collectRequestParams: # [array] Collect parameters - Optional collect request p
    - name: # [string] Name - Parameter name
      value: # [string] Value - JavaScript expression to compute the parameter's

  # -----------------------------------------------------
```

```
# -------------- if collectMethod is post ---------------

collectRequestParams: # [array] Collect parameters - Optional collect request |
  - name: # [string] Name - Parameter name.
    value: # [string] Value - JavaScript expression to compute the parameter's

# -------------------------------------------------------


# -------------- if collectMethod is post_with_body ---------------

collectBody: # [string] Collect POST body - Template for POST body to send witl

# -------------------------------------------------------


# -------------- if collectMethod is other ---------------

collectVerb: # [string] Collect verb - DIY HTTP verb to use for the collect op
collectBody: # [string] Collect body - Template for body to send with the Coll
collectRequestParams: # [array] Collect parameters - Optional collect request |
  - name: # [string] Name - Parameter name.
    value: # [string] Value - JavaScript expression to compute the parameter's

# -------------------------------------------------------

collectRequestHeaders: # [array] Collect headers - Optional collect request hea
  - name: # [string] Name - Header Name
    value: # [string] Value - JavaScript expression to compute the header's va
pagination: # [object]
  type: # [string] Pagination - Select collect pagination scheme

  # -------------- if type is response_body ---------------

  attribute: # [array,string] Response attribute - The name of the attribute i
  maxPages: # [number] Max pages - The maximum number of pages to retrieve, se

  # -------------------------------------------------------


  # -------------- if type is response_header_link ---------------
```

```
  nextRelationAttribute: # [string] Next page relation name - Relation name us
  curRelationAttribute: # [string] Current page relation name - Optional relat
  maxPages: # [number] Max pages - The maximum number of pages to retrieve, se

  # ------------------------------------------------------


  # -------------- 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 reques
  offsetSpacer: # [null]
  limitField: # [string] Limit field name - Query string parameter to set the
  limit: # [number] Limit - Maximum number of records to collect per request.
  limitSpacer: # [null]
  totalRecordField: # [string] Total record count field name - Identifies the
  maxPages: # [number] Max pages - The maximum number of pages to retrieve. Se
  zeroIndexed: # [boolean] Zero-based index - Toggle to Yes to indicate that t

  # ------------------------------------------------------


  # -------------- if type is request_page ---------------

  pageField: # [string] Page number field name - Query string parameter that s
  page: # [number] Starting page number - Page number from which to start requ
  offsetSpacer: # [null]
  sizeField: # [string] Page size field name - Query string parameter to set t
  size: # [number] Page size - Maximum number of records to collect per page.
  limitSpacer: # [null]
  totalPageField: # [string] Total page count field name - The name of the att
  totalRecordField: # [string] Total record count field name - Identifies the
  maxPages: # [number] Max pages - The maximum number of pages to retrieve. Se
  zeroIndexed: # [boolean] zero-based index - Toggle to Yes to indicate that t

  # ------------------------------------------------------

authentication: # [string] [required] Authentication - Authentication method f

# -------------- 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 s

# ------------------------------------------------------


# -------------- if authentication is login ---------------

loginUrl: # [string] Login URL - URL to use for login API call. This call is e
username: # [string] Username - Login username
password: # [string] Password - Login password
loginBody: # [string] POST body - Template for POST body to send with login re
tokenRespAttribute: # [string] Token attribute - Path to token attribute in lo
authHeaderExpr: # [string] Authorize expression - JavaScript expression to com
authRequestHeaders: # [array] Authentication headers - Optional authentication
  - name: # [string] Name - Header name.
    value: # [string] Value - JavaScript expression to compute the header's va

# ------------------------------------------------------


# -------------- if authentication is loginSecret ---------------

loginUrl: # [string] Login URL - URL to use for login API call, this call is e
credentialsSecret: # [string] Credentials secret - Select or create a stored s
loginBody: # [string] POST body - Template for POST body to send with login re
tokenRespAttribute: # [string] Token attribute - Path to token attribute in lo
authHeaderExpr: # [string] Authorize expression - JavaScript expression to com
authRequestHeaders: # [array] Authentication headers - Optional authentication
  - name: # [string] Name - Header name.
    value: # [string] Value - JavaScript expression to compute the parameter's

# ------------------------------------------------------


# -------------- if authentication is oauth ---------------

loginUrl: # [string] Login URL - URL to use for the OAuth API call. This call
```

```
  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 s
  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] Request timeout (secs) - HTTP request inactivity timeout, u
  useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS t
  disableTimeFilter: # [boolean] Disable time filter - Used to disable collector
  safeHeaders: # [array of strings] Safe headers - List of headers that are safe

# ------------------------------------------------------


# -------------- if type is s3 ---------------

conf: # [object] [required]
  outputName: # [string] Auto-populate from - The name of the predefined Destinat
  bucket: # [string] S3 bucket - S3 Bucket from which to collect data.
```

```
  region: # [string] Region - Region from which to retrieve data.
  path: # [string] Path - The directory from which to collect data. Templating is
  extractors: # [array] Path extractors - Allows using template tokens as contex
    - key: # [string] Token - A token from the template path, e.g.: epoch
      expression: # [string] Extractor expression - JS expression that receives
  awsAuthenticationMethod: # [string] Authentication Method - AWS authentication

  # -------------- if awsAuthenticationMethod is manual ---------------

  awsApiKey: # [string] Access key - Access key. If not present, will fall back
  awsSecretKey: # [string] Secret key - Secret key. If not present, will fall ba

  # --------------------------------------------------------


  # -------------- if awsAuthenticationMethod is secret ---------------

  awsSecret: # [string] Secret key pair - Select or create a stored secret that

  # --------------------------------------------------------

  endpoint: # [string] Endpoint - S3 service endpoint. If empty, the endpoint wi
  signatureVersion: # [string] Signature version - Signature version to use for
  enableAssumeRole: # [boolean] Enable Assume Role - Use Assume Role credentials
  assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the r
  assumeRoleExternalId: # [string] External ID - External ID to use when assumin
  recurse: # [boolean] Recursive - Whether to recurse through subdirectories.
  maxBatchSize: # [number] Max Batch Size (objects) - Maximum number of metadata
  reuseConnections: # [boolean] Reuse Connections - Whether to reuse connections
  rejectUnauthorized: # [boolean] Reject Unauthorized Certificates - Whether to
  verifyPermissions: # [boolean] Verify bucket permissions - Disable if you can

# --------------------------------------------------------


# -------------- if type is script ---------------

conf: # [object] [required]
  discoverScript: # [string] Discover Script - Script to discover what to collec
  collectScript: # [string] [required] Collect Script - Script to run to perform
  shell: # [string] Shell - Shell to use to execute scripts.

# --------------------------------------------------------
```

```
# -------------- if type is splunk ---------------

conf: # [object] [required]
  searchHead: # [string] [required] Search head - Search Head base URL, can be ex
  search: # [string] Search - Enter Splunk search here. For example: 'index=myApp
  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 pa
    - name: # [string] Name - Parameter name
      value: # [string] Value - JavaScript expression to compute the parameter's
  collectRequestHeaders: # [array] Extra headers - Optional collect request head
    - name: # [string] Name - Header Name
      value: # [string] Value - JavaScript expression to compute the header's va
  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 re
  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 req
    tokenRespAttribute: # [string] Token Attribute - Path to token attribute in log
    authHeaderExpr: # [string] Authorize Expression - JavaScript expression to comp

    # -------------------------------------------------------

    timeout: # [number] Request Timeout (secs) - HTTP request inactivity timeout, u
    useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS f
    disableTimeFilter: # [boolean] Disable time filter - Used to disable collector

  # -------------------------------------------------------

  destructive: # [boolean] Destructive - If set to Yes, the collector will delete a
input: # [object]
  type: # [string]
  breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event br
  staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of time
  sendToRoutes: # [boolean] Send to Routes - If set to Yes, events will be sent to

  # -------------- if sendToRoutes is true ---------------

  pipeline: # [string] Pre-Processing Pipeline - Pipeline to process results before

  # -------------------------------------------------------


  # -------------- if sendToRoutes is false ---------------

  pipeline: # [string] Pipeline - Pipeline to process results.
  output: # [string] Destination - Destination to send results to.

  # -------------------------------------------------------

  preprocess: # [object]
    disabled: # [boolean] Disabled - Enable Custom Command
    command: # [string] Command - Command to feed the data through (via stdin) and
```

```
    command: # [string] Command - Command to feed the data through (via stdin) and
    args: # [array of strings] Arguments - Arguments
  throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to throtti
  metadata: # [array] Fields - Fields to add to events from this input.
    - name: # [string] Name - Field name
      value: # [string] Value - JavaScript expression to compute field's value, en
type: # [string] Job type - Job type.
ttl: # [string] Time to live - Time to keep the job's artifacts on disk after job c
removeFields: # [array of strings] Remove Discover fields - List of fields to remov
resumeOnBoot: # [boolean] Resume job on boot - Resumes the ad hoc job if a failure
environment: # [string] Environment - Optionally, enable this config only on a spec
schedule: # [object] Schedule - Configuration for a scheduled job.
  enabled: # [boolean] Enabled - Determines whether or not this schedule is enabled

  # -------------- if enabled is true ---------------

  cronSchedule: # [string] Cron schedule - A cron schedule on which to run this jol
  maxConcurrentRuns: # [number] Max concurrent runs - The maximum number of instanc
  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] Max task reschedule - Max number of times a task
    logLevel: # [string] Log Level - Level at which to set task logging.
    jobTimeout: # [string] Job timeout - Maximum time the job is allowed to run (e
    mode: # [string] Mode - Job run mode. Preview will either return up to N matchi

    # -------------- if mode is list ---------------

    discoverToRoutes: # [boolean] Send to Routes - If true, send discover results t

    # -----------------------------------------------------



    # -------------- if mode is preview ---------------

    capture: # [object] Capture Settings
      duration: # [number] Capture Time (sec) - Amount of time to keep capture open
      maxEvents: # [number] Capture Up to N Events - Maximum number of events to ca
      level: # [string] Where to capture

    # -----------------------------------------------------
```

```
      "

      timeRangeType: # [string] Time range - Time range for scheduled job.
      earliest: # [number,string] Earliest - Earliest time, in local time.
      latest: # [number,string] Latest - Latest time, in local time.
      timestampTimezone: # [string] Range Timezone - Timezone to use for Earliest an
      expression: # [string] Filter - A filter for tokens in the provided collect pa
      minTaskSize: # [string] Lower task bundle size - Limits the bundle size for sm
      maxTaskSize: # [string] Upper task bundle size - Limits the bundle size for fi

    # ------------------------------------------------------

  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in Str
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
    conf: # [object] Executor-specific settings.
  type: # [string] Job type - Job type.
  ttl: # [string] Time to live - Time to keep the job's artifacts on disk after job
  removeFields: # [array of strings] Remove Discover fields - List of fields to remo
  resumeOnBoot: # [boolean] Resume job on boot - Resumes the ad hoc job if a failure
  environment: # [string] Environment - Optionally, enable this config only on a spe
  schedule: # [object] Schedule - Configuration for a scheduled job.
    enabled: # [boolean] Enabled - Determines whether or not this schedule is enable

    # -------------- if enabled is true ---------------

    cronSchedule: # [string] Cron schedule - A cron schedule on which to run this jo
    maxConcurrentRuns: # [number] Max concurrent runs - The maximum number of instan
    skippable: # [boolean] Skippable - Skippable jobs can be delayed, up to their ne
    run: # [object] Run settings

      # -------------- if type is collection ---------------

      rescheduleDroppedTasks: # [boolean] Reschedule tasks - Reschedule tasks that f
      maxTaskReschedule: # [number] Max task reschedule - Max number of times a task
      logLevel: # [string] Log Level - Level at which to set task logging.
      jobTimeout: # [string] Job timeout - Maximum time the job is allowed to run (e
```

The `workerAffinity` internal parameter defaults to `false`. Cribl automatically sets this to `true` on certain jobs, specifying that collection tasks will be both created and run by the same Edge Node.

```
      discoverToRoutes: # [boolean] Send to Routes - If true, send discover results
```

```
    discoverToRoutes: # [boolean] Send to Routes - If true, send discover results

    # -------------------------------------------------------


    # -------------- if mode is preview ---------------

    capture: # [object] Capture Settings
      duration: # [number] Capture Time (sec) - Amount of time to keep capture oper
      maxEvents: # [number] Capture Up to N Events - Maximum number of events to ca
      level: # [string] Where to capture

    # -------------------------------------------------------

    timeRangeType: # [string] Time range - Time range for scheduled job.
    earliest: # [number,string] Earliest - Earliest time, in local time.
    latest: # [number,string] Latest - Latest time, in local time.
    timestampTimezone: # [string] Range Timezone - Timezone to use for Earliest and
    expression: # [string] Filter - A filter for tokens in the provided collect pa
    minTaskSize: # [string] Lower task bundle size - Limits the bundle size for sma
    maxTaskSize: # [string] Upper task bundle size - Limits the bundle size for fi

  # -----------------------------------------------------------

streamtags: # [array of strings] Tags - Add tags for filtering and grouping in Stre
```

15.4.8. 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.

$CRIBL_HOME/default/cribl/job-limits.yml

```
concurrentJobLimit: # [number; default: 10, minimum: 1] Concurrent job limit - The t
concurrentSystemJobLimit: # [number; default: 10, minimum: 1] Concurrent system job
concurrentScheduledJobLimit: # [number, default: -2] Concurrent scheduled job limit
concurrentTaskLimit: # [number; default: 2, minimum: 1] Concurrent task limit - The
concurrentSystemTaskLimit: # [number; default: 1, minimum: 1] Concurrent system task
maxTaskPerc: # [number; default: 0.5, minimum: 0, maximum: 1] Max task usage percent
taskPollTimeoutMs: # [number; default: 60000, minimum: 10000] Task poll timeout - Th
jobArtifactsReaperPeriod: # [string; default: 30 min.] Artifact reaper period - Time
finishedJobArtifactsLimit: # [number; default: 100 items, minimum: 0] Finished job a
finishedTaskArtifactsLimit: # [number; default: 500 items, minimum: 0] Finished task
taskManifestFlushPeriodMs: # [number; default: 100, minimum: 100, maximum: 10000] Ma
taskManifestMaxBufferSize: # [number; default: 1000, minimum: 100, maximum: 10000] M
taskManifestReadBufferSize: # [string; default: 4 KB] Manifest reader buffer size -
schedulingPolicy: # [string; default: LeastInFlight] Job dispatching - The method by
jobTimeout: # [string; default: 0] Job timeout - Maximum time a job is allowed to ru
taskHeartbeatPeriod: # [number; default: 60, minimum: 60] Task heartbeat period - Th
```

15.4.9. 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.

$CRIBL_HOME/local/cribl/leader.yml

```
distributed:
  mode: # [string] One of: master | worker | single | edge | managed-edge.
  master: # [object]
    disableSNIRouting: # [boolean] Whether SNI routing is enabled. See (Environment
    host: # [string] Instance host address.
    port: # [number] Instance post number.
    tls: # [object]
      disabled: # [boolean]
      certificateName: # [string] TLS certificate name.
      rejectUnauthorized: # [boolean] False if ignoring untrusted certificates.
      requestCert: # [boolean] Whether certificates are required and validated.
      privKeyPath: # [string] Path to private key file. Can reference $ENV_VARS.
      certPath: # [string] Path to certificate file. Can reference $ENV_VARS.
      caPath: # [string] Path to CA certificate file. Can reference $ENV_VARS.
      commonNameRegex: # [any] Allowed common names in peer certificates' subject (r
      # Available values: 'TLSv1.3' | 'TLSv1.2' | 'TLSv1.1' | 'TLSv1'
      minVersion: # [string] Minimum TLS version.
      maxVersion: # [string] Maximum TLS version.
      servername: # [string] Server name.
    ipWhitelistRegex: # [string] IP addresses allowed to send data (regex).
    maxActiveCxn: # [number] Max number of active connections allowed per Worker Proc
    authToken: # [string] Token used for communication between Leader and managed no
    compression: # [string] Codec used to compress persisted data. One of: none | gz
    connectionTimeout: # [number] Maximum time to wait for a connection to complete
    writeTimeout: # [number] Time (ms) to wait for a write to complete before assumi
    configBundles: # [object]
      remoteUrl: # [string] Bucket to use for remote bundle storage, in s3://${bucke
  group: # [string] Group the managed node belongs to.
  envRegex: # [string] Regex used to filter env variable names that can be sent by ma
  tags: # [array of strings] Tags associated with managed node, can be used by Leade
```

15.4.10. Licenses.Yml

`licenses.yml` maintains a list of Cribl Edge licenses.

$CRIBL_HOME/default/cribl/licenses.yml

```
licenses: # [array of string] - list of licenses
```

15.4.11. 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.

$CRIBL_HOME/default/cribl/limits.yml

```
samples: # [object] Samples
  maxSize: # [string; maximum: 3 MB; default: 256 KB] Max sample size - Maximum file
maxMetrics: # [number; default: 1000000] Total allowed number of metric series, conf
minFreeSpace: # [string; default: 5 GB] Min free disk space - The minimum amount of
maxPQSize: # [string; default: 1 TB] Maximum persistent queue size per Worker Proces
metricsGCPeriod: # [string; default: 60s] Metrics GC period - The interval on which
metricsMaxCardinality: # [number; default: 1000] Metrics cardinality limit - The sys
metricsMaxDiskSpace: # [string; default: 64GB] Metrics max disk space - Maximum allo
metricsWorkerIdBlacklist: # [array of strings] Metrics worker tracking - List of met
metricsNeverDropList: # [array of strings] Metrics never-drop list - List of metric
metricsFieldsBlacklist: # [array of strings] Disable field metrics - List of event f
metricsDirectory: # [string] Metrics directory - Directory in which to store metrics
cpuProfileTTL: # [string; default: 30m] CPU profile TTL - The time-to-live for colle
eventsMetadataSources: # [array of strings] Event metadata sources - List of event m
edgeMetricsMode: # [string] Metrics to send from Edge Nodes - Choose a set of metric
kvStoreEnableWAL: # [boolean; default: false] Use Write-Ahead Log (WAL) mode — When
kvStoreWALMaxBytes: # [number; default: 5242880] Maximum size of the WAL file in byt
kvStoreServicePeriod: # [number; enabled default: 60000; disabled default: 10000] In
bundleDownloadTimeoutSeconds: # [number; default: 0] Maximum time a Cribl Stream Wor

# -------------- if edgeMetricsMode is custom ---------------

edgeMetricsCustomExpression: # [string] Metrics expression - JavaScript expression t

# -----------------------------------------------------

enableMetricsPersistence: # [boolean; default: true] Persist metrics - Set this to f
```

15.4.12. Logger.Yml

`logger.yml` maintains logging levels and redactions, per channel. In the UI, you can configure these at **Settings** > **Global** > **System** > **Logging**.

$CRIBL_HOME/default/cribl/logger.yml

```
redactFields: # [array of strings] - list of fields to redact
redactLabel: # [string] - redact label
channels: # [object] Logger channels as logger ID/log level pairs. Log levels: error
  DEFAULT: info
  input:DistMaster: debug
  output:DistWorker: debug
```

15.4.13. Mappings.Yml

Mapping ruleset configurations are stored in `$CRIBL_HOME/default/cribl/mappings.yml`.

$CRIBL_HOME/default/cribl/mappings.yml

```
mapping_ruleset_id: # [object]
  conf: # [object]
    functions: # [array] Functions - List of functions to pass data through
  active: # [boolean]
```

15.4.14. Messages.Yml

`messages.yml` stores messages displayed in the UI's `Messages` fly-out.

$CRIBL_HOME/local/cribl/logger.yml

```
message_id: # [object]
  severity: # [string] Severity
  title: # [string] Title
  text: # [string] Text
  time: # [number] Occurrence Time
  group: # [string] Group
  metadata: # [array]
```

15.4.15. Outputs.Yml

`outputs.yml` contains configuration settings for Cribl Edge Destinations.

$CRIBL_HOME/default/cribl/outputs.yml

```
outputs: # [object]
  default_output: # [object]
    type: # [string] Output Type
    defaultId: # [string,null] Default Output ID - ID of the default output. This wi
    pipeline: # [string] Pipeline - Pipeline to process data before sending out to t
    systemFields: # [array of strings] System fields - Set of fields to automatical
    environment: # [string] Environment - Optionally, enable this config only on a s
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in S
  webhook_output: # [object]
    type: # [string] Output Type
    url: # [string] URL - URL to send events to.
    method: # [string] Method - The method to use when sending events. Defaults to P
    format: # [string] Format - Specifies how to format events before sending out. D

    # -------------- if format is custom ---------------

    customSourceExpression: # [string] Source expression - Expression to evaluate as
    customDropWhenNull: # [boolean] Drop when null - Whether or not to drop events w
    customEventDelimiter: # [string] Event delimiter - Delimiter string to insert be
    customContentType: # [string] Content type - Content type to use for request. De

    # --------------------------------------------------------

    concurrency: # [number] Request concurrency - Maximum number of ongoing requests
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the re
    maxPayloadEvents: # [number] Max events per request - Max number of events to in
    compress: # [boolean] Compress - Whether to compress the payload body before sen
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are no
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. S
    extraHttpHeaders: # [array] Extra HTTP headers - Extra HTTP headers.
      - name: # [string] Name - Field name
        value: # [string] Value - Field value
    useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS lo
    failedRequestLoggingMode: # [string] Failed request logging mode - Determines wh
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe t
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or qu

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each queu
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the queu
```

```
  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 - Whether to block or drop even
  pqControls: # [object]

  # ------------------------------------------------------

  authType: # [string] Authentication type - The authentication method to use for

  # -------------- if authType is basic ---------------

  username: # [string] Username - Username for Basic authentication
  password: # [string] Password - Password for Basic authentication

  # ------------------------------------------------------


  # -------------- 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 tha

  # ------------------------------------------------------


  # -------------- if authType is textSecret ---------------

  textSecret: # [string] Token (text secret) - Select or create a stored text secr

  # ------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to t
  systemFields: # [array of strings] System fields - Set of fields to automatically
  environment: # [string] Environment - Optionally, enable this config only on a s
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in S
devnull_output: # [object]
  type: # [string] Output Type
```

```
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to tl
  systemFields: # [array of strings] System fields - Set of fields to automatically
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in St
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 - Use load-balanced destinations
  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 certs that are n
    servername: # [string] Server name (SNI) - Server name for the SNI (Server Name
    certificateName: # [string] Certificate name - The name of the predefined certi
    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 - Minimum TLS version to use when co
    maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when co

    # ------------------------------------------------------

  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or que

  # ------------------------------------------------------


  # -------------- if protocol is udp ---------------

  host: # [string] Address - The hostname of the receiver
  port: # [number] Port - The port to connect to on the provided host
  maxRecordSize: # [number] Max Record Size - Maximum size of syslog messages. If n

  # ------------------------------------------------------
```

```
  facility: # [number] Facility - Default value for message facility, will be overw
  severity: # [number] Severity - Default value for message severity, will be overw
  appName: # [string] App Name - Default value for application name, will be overwr
  messageFormat: # [string] Message Format - The syslog message format depending or
  timestampFormat: # [string] Timestamp Format - Timestamp format to use when seria
  throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to thrott
  octetCountFraming: # [boolean] Octet count framing - When enabled, messages will
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Set of fields to automatically
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in Sp
splunk_output: # [object]
  type: # [string] Output Type
  host: # [string] Address - The hostname of the receiver
  port: # [number] [required] Port - The port to connect to on the provided host
  nestedFields: # [string] Nested field serialization - Specifies how to serialize
  throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to thrott
  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 certs that are n
    servername: # [string] Server name (SNI) - Server name for the SNI (Server Name
    certificateName: # [string] Certificate name - The name of the predefined certi
    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 - Minimum TLS version to use when co
    maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when co

    # ----------------------------------------------------

  enableMultiMetrics: # [boolean] Output multiple metrics - Output metrics in multi
  enableACK: # [boolean] Minimize in-flight data loss - Check if indexer is shuttin
  maxS2Sversion: # [string] Max S2S version - The highest S2S protocol version to a
  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or que

  # -------------- if onBackpressure is queue ---------------
```

```
  pqMaxFileSize: # [string] Max file size - The maximum size to store in each queu
  pqMaxSize: # [string] Max queue size - The maximum amount of disk space the queu
  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 - Whether to block or drop even
  pqControls: # [object]

  # ------------------------------------------------------

  authType: # [string] Authentication method - Enter a token directly, or provide a
  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 ---------------


  # ------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Set of fields to automatically
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in Sp
splunk_lb_output: # [object]
  type: # [string] Output Type
  dnsResolvePeriodSec: # [number] DNS resolution period (seconds) - Re-resolve any
  loadBalanceStatsPeriodSec: # [number] Load balance stats period (seconds) - How f
  maxConcurrentSenders: # [number] Max connections - Maximum number of concurrent c
  nestedFields: # [string] Nested field serialization - Specifies how to serialize
  throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to thrott
  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 certs that are n
```

```
  servername: # [string] Server name (SNI) - Server name for the SNI (Server Nam
  certificateName: # [string] Certificate name - The name of the predefined cert
  caPath: # [string] CA certificate path - Path on client in which to find CA ce
  privKeyPath: # [string] Private key path (mutual auth) - Path on client in whi
  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 - Minimum TLS version to use when c
  maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when c

  # ------------------------------------------------------

enableMultiMetrics: # [boolean] Output multiple metrics - Output metrics in mult
enableACK: # [boolean] Minimize in-flight data loss - Check if indexer is shutti
maxS2Sversion: # [string] Max S2S version - The highest S2S protocol version to
onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or qu

# -------------- if onBackpressure is queue ---------------

pqMaxFileSize: # [string] Max file size - The maximum size to store in each queu
pqMaxSize: # [string] Max queue size - The maximum amount of disk space the queu
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 - Whether to block or drop even
pqControls: # [object]

# ------------------------------------------------------

indexerDiscovery: # [boolean] Indexer Discovery - Automatically discover indexer

# -------------- if indexerDiscovery is true ---------------

indexerDiscoveryConfigs: # [object]  - List of configurations to set up indexer
  site: # [string] [required] Site - Clustering site of the indexers from where
  masterUri: # [string] Cluster Manager URI - Full URI of Splunk cluster Manager
  refreshIntervalSec: # [number] [required] Refresh Period - Time interval in se
  authType: # [string] Authentication method - Enter a token directly, or provid
  authToken: # [string] Auth token - Authentication token required to authentica

  # -------------- if authType is manual ---------------


  # ------------------------------------------------------
```

```
  textSecret: # [string] Auth token (text secret) - Select or create a stored te

  # -------------- if authType is secret ---------------


  # -------------------------------------------------------


# -----------------------------------------------------


# -------------- if indexerDiscovery is false ---------------

excludeSelf: # [boolean] Exclude current host IPs - Exclude all IPs of the curre
hosts: # [array] Destinations - Set of Splunk indexers to load-balance data to.
  - host: # [string] Address - The hostname of the receiver.
    port: # [number] Port - The port to connect to on the provided host.
    tls: # [string] TLS - Whether to inherit TLS configs from group setting or d
    servername: # [string] TLS Servername - Servername to use if establishing a
    weight: # [number] Load Weight - The weight to use for load-balancing purpos
dnsResolvePeriodSec: # [number] DNS resolution period (seconds) - Re-resolve any
loadBalanceStatsPeriodSec: # [number] Load balance stats period (seconds) - How
maxConcurrentSenders: # [number] Max connections - Maximum number of concurrent

# -----------------------------------------------------

authType: # [string] Authentication method - Enter a token directly, or provide
authToken: # [string] Auth token - Shared secret token to use when establishing

# -------------- if authType is manual ---------------


# -----------------------------------------------------

textSecret: # [string] Auth token (text secret) - Select or create a stored text

# -------------- if authType is secret ---------------


# -----------------------------------------------------

pipeline: # [string] Pipeline - Pipeline to process data before sending out to t
systemFields: # [array of strings] System fields - Set of fields to automatical
```

```
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in Sp
splunk_hec_output: # [object]
  type: # [string] Output Type
  loadBalanced: # [boolean] Load balancing - Use load-balanced destinations

  # -------------- if loadBalanced is false ---------------

  url: # [string] Splunk HEC Endpoint - URL to a Splunk HEC endpoint to send events
  useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS loo

  # ------------------------------------------------------



  # -------------- if loadBalanced is true ---------------

  excludeSelf: # [boolean] Exclude current host IPs - Exclude all IPs of the curren
  urls: # [array] Splunk HEC Endpoints
    - url: # [string] HEC Endpoint - URL to a Splunk HEC endpoint to send events to
      weight: # [number] Load Weight - The weight to use for load-balancing purpose
  dnsResolvePeriodSec: # [number] DNS resolution period (seconds) - Re-resolve any
  loadBalanceStatsPeriodSec: # [number] Load balance stats period (seconds) - How f

  # ------------------------------------------------------



  nextQueue: # [string] Next Processing Queue - Which Splunk processing queue to se
  tcpRouting: # [string] Default _TCP_ROUTING - Set the value of _TCP_ROUTING for e
  concurrency: # [number] Request concurrency - Maximum number of ongoing requests
  maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the req
  maxPayloadEvents: # [number] Max events per request - Max number of events to inc
  compress: # [boolean] Compress - Whether to compress the payload body before send
  rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are not
  timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
  extraHttpHeaders: # [array] Extra HTTP headers - Extra HTTP headers.
    - name: # [string] Name - Field name
      value: # [string] Value - Field value
  failedRequestLoggingMode: # [string] Failed request logging mode - Determines whi
  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 - Enter a token directly, or provide a

  # -------------- 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

  # ------------------------------------------------------

  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or qu

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] Max file size - The maximum size to store in each queu
  pqMaxSize: # [string] Max queue size - The maximum amount of disk space the queu
  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 - Whether to block or drop even
  pqControls: # [object]

  # ------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to t
  systemFields: # [array of strings] System fields - Set of fields to automatically
  environment: # [string] Environment - Optionally, enable this config only on a s
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in S
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] Port - The port to connect to on the provided host

  # ------------------------------------------------------

  # -------------- if loadBalanced is true ---------------
```

```
excludeSelf: # [boolean] Exclude current host IPs - Exclude all IPs of the curre
hosts: # [array] Destinations - Set of hosts to load-balance data to.
  - host: # [string] Address - The hostname of the receiver.
    port: # [number] Port - The port to connect to on the provided host.
    tls: # [string] TLS - Whether to inherit TLS configs from group setting or d
    servername: # [string] TLS Servername - Servername to use if establishing a
    weight: # [number] Load Weight - The weight to use for load-balancing purpos
dnsResolvePeriodSec: # [number] DNS resolution period (seconds) - Re-resolve any
loadBalanceStatsPeriodSec: # [number] Load balance stats period (seconds) - How
maxConcurrentSenders: # [number] Max connections - Maximum number of concurrent

# ------------------------------------------------------

compression: # [string] Compression - Codec to use to compress the data before s
throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to thrott
tls: # [object] TLS settings (client side)
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
  servername: # [string] Server name (SNI) - Server name for the SNI (Server Nam
  certificateName: # [string] Certificate name - The name of the predefined cert
  caPath: # [string] CA certificate path - Path on client in which to find CA ce
  privKeyPath: # [string] Private key path (mutual auth) - Path on client in whi
  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 - Minimum TLS version to use when c
  maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when c

  # ------------------------------------------------------

connectionTimeout: # [number] Connection Timeout - Amount of time (milliseconds)
writeTimeout: # [number] Write Timeout - Amount of time (milliseconds) to wait f
tokenTTLMinutes: # [number] Auth Token TTL minutes - The number of minutes befor
onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or qu

# -------------- if onBackpressure is queue ---------------

pqMaxFileSize: # [string] Max file size - The maximum size to store in each queu
pqMaxSize: # [string] Max queue size - The maximum amount of disk space the queu
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 - Whether to block or drop even
  pqControls: # [object]

  # ------------------------------------------------------

  authType: # [string] Authentication method - Enter a token directly, or provide a
  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 ---------------


  # ------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Set of fields to automatically
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in Sp
wavefront_output: # [object]
  type: # [string] Output Type
  authType: # [string] Authentication method - Enter a token directly, or provide a

  # -------------- 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] Request concurrency - Maximum number of ongoing requests
```

```
  maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the re
  maxPayloadEvents: # [number] Max events per request - Max number of events to in
  compress: # [boolean] Compress - Whether to compress the payload body before sen
  rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are no
  timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. S
  extraHttpHeaders: # [array] Extra HTTP headers - Extra HTTP headers.
    - name: # [string] Name - Field name
      value: # [string] Value - Field value
  useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS lo
  failedRequestLoggingMode: # [string] Failed request logging mode - Determines wh
  safeHeaders: # [array of strings] Safe headers - List of headers that are safe t
  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or qu

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] Max file size - The maximum size to store in each queu
  pqMaxSize: # [string] Max queue size - The maximum amount of disk space the queu
  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 - Whether to block or drop even
  pqControls: # [object]

  # ---------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to t
  systemFields: # [array of strings] System fields - Set of fields to automatical
  environment: # [string] Environment - Optionally, enable this config only on a s
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in S
signalfx_output: # [object]
  type: # [string] Output Type
  authType: # [string] Authentication method - Enter a token directly, or provide

  # -------------- if authType is manual ---------------

  token: # [string] Auth token - SignalFx API access token (see [here](https://doc

  # ---------------------------------------------------------


  # -------------- 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"
  concurrency: # [number] Request concurrency - Maximum number of ongoing requests
  maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the re
  maxPayloadEvents: # [number] Max events per request - Max number of events to in
  compress: # [boolean] Compress - Whether to compress the payload body before sen
  rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are no
  timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. S
  extraHttpHeaders: # [array] Extra HTTP headers - Extra HTTP headers.
    - name: # [string] Name - Field name
      value: # [string] Value - Field value
  useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS lo
  failedRequestLoggingMode: # [string] Failed request logging mode - Determines wh
  safeHeaders: # [array of strings] Safe headers - List of headers that are safe t
  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or qu

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] Max file size - The maximum size to store in each queu
  pqMaxSize: # [string] Max queue size - The maximum amount of disk space the queu
  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 - Whether to block or drop even
  pqControls: # [object]

  # ------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to t
  systemFields: # [array of strings] System fields - Set of fields to automatical
  environment: # [string] Environment - Optionally, enable this config only on a s
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in S
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
  addIdToStagePath: # [boolean] Add output ID - Append output's ID to staging loca
  removeEmptyDirs: # [boolean] Remove staging dirs - Remove empty staging directo

  # -------------- if removeEmptyDirs is true ---------------
```

```
  emptyDirCleanupSec: # [number] Staging cleanup period - How often (secs) to clear

  # ------------------------------------------------------

  partitionExpr: # [string] Partitioning expression - JS expression defining how f
  format: # [string] Data format - Format of the output data.

  # -------------- if format is json ---------------

  compress: # [string] Compress - Choose data compression format to apply before m

  # ------------------------------------------------------


  # -------------- if format is parquet ---------------

  parquetSchema: # [string] Parquet schema - Select a Parquet schema. New schemas
  parquetRowGroupSize: # [string] Row group size - Ideal memory size for row group
  parquetPageSize: # [string] Page size - Ideal memory size for page segments. E.g
  spacer: # [null]
  parquetVersion: # [string] Parquet version - Determines which data types are supp
  parquetDataPageVersion: # [string] Data page version - Serialization format of da
  shouldLogInvalidRows: # [boolean] Log invalid rows - Output up to 20 unique rows

  # ------------------------------------------------------

  baseFileName: # [string] File name prefix expression - JavaScript expression to
  fileNameSuffix: # [string] File name suffix expression - JavaScript expression to
  maxFileSizeMB: # [number] Max file size (MB) - Maximum uncompressed output file s
  maxFileOpenTimeSec: # [number] Max file open time (Sec) - Maximum amount of time
  maxFileIdleTimeSec: # [number] Max file idle time (Sec) - Maximum amount of time
  maxOpenFiles: # [number] Max open files - Maximum number of files to keep open c
  onBackpressure: # [string] Backpressure behavior - Whether to block or drop event
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Set of fields to automatically
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in S
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.
  awsAuthenticationMethod: # [string] Authentication method - AWS authentication m
```

```
# -------------- if awsAuthenticationMethod is manual ---------------

awsApiKey: # [string] Access key - Access key

# -------------------------------------------------------


# -------------- if awsAuthenticationMethod is secret ---------------

awsSecret: # [string] Secret key pair - Select or create a stored secret that re

# -------------------------------------------------------

awsSecretKey: # [string] Secret key - Secret key
endpoint: # [string] Endpoint - S3 service endpoint. If empty, defaults to AWS' r
signatureVersion: # [string] Signature version - Signature version to use for sig
reuseConnections: # [boolean] Reuse connections - Whether to reuse connections be
rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to re
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
stagePath: # [string] [required] Staging location - Filesystem location in which
addIdToStagePath: # [boolean] Add output ID - Append output's ID to staging locat
removeEmptyDirs: # [boolean] Remove staging dirs - Remove empty staging director

# -------------- if removeEmptyDirs is true ---------------

emptyDirCleanupSec: # [number] Staging cleanup period - How often (secs) to clean

# -------------------------------------------------------

destPath: # [string] [required] Key prefix - Prefix to append to files before up
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

# -------------- if serverSideEncryption is aws:kms ---------------

kmsKeyId: # [string] KMS key ID - ID or ARN of the KMS customer managed key to us

# -------------------------------------------------------

partitionExpr: # [string] Partitioning expression - JS expression defining how f
```

```
  partitionExpr: # [string] Partitioning expression - JS expression defining how t
  format: # [string] Data format - Format of the output data.

  # -------------- if format is json ---------------

  compress: # [string] Compress - Choose data compression format to apply before m

  # ------------------------------------------------------


  # -------------- if format is parquet ---------------

  parquetSchema: # [string] Parquet schema - Select a Parquet schema. New schemas
  parquetRowGroupSize: # [string] Row group size - Ideal memory size for row group
  parquetPageSize: # [string] Page size - Ideal memory size for page segments. E.g
  spacer: # [null]
  parquetVersion: # [string] Parquet version - Determines which data types are supp
  parquetDataPageVersion: # [string] Data page version - Serialization format of da
  shouldLogInvalidRows: # [boolean] Log invalid rows - Output up to 20 unique rows

  # ------------------------------------------------------

  baseFileName: # [string] File name prefix expression - JavaScript expression to
  fileNameSuffix: # [string] File name suffix expression - JavaScript expression to
  maxFileSizeMB: # [number] Max file size (MB) - Maximum uncompressed output file s
  maxFileOpenTimeSec: # [number] Max file open time (Sec) - Maximum amount of time
  maxFileIdleTimeSec: # [number] Max file idle time (Sec) - Maximum amount of time
  maxOpenFiles: # [number] Max open files - Maximum number of files to keep open c
  onBackpressure: # [string] Backpressure behavior - Whether to block or drop even
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Set of fields to automatically
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in St
azure_blob_output: # [object]
  type: # [string] Output Type
  containerName: # [string] Container name - A container organizes a set of blobs,
  createContainer: # [boolean] Create container - Creates the configured container
  destPath: # [string] [required] Blob prefix - Root directory prepended to path bl
  stagePath: # [string] [required] Staging location - Filesystem location in which
  addIdToStagePath: # [boolean] Add output ID - Append output's ID to staging loca
  removeEmptyDirs: # [boolean] Remove staging dirs - Remove empty staging director

  # -------------- if removeEmptyDirs is true ---------------
```

```
emptyDirCleanupSec: # [number] Staging cleanup period - How often (secs) to clea

# -------------------------------------------------------

partitionExpr: # [string] Partitioning expression - JS expression defining how f
format: # [string] Data format - Format of the output data.

# -------------- if format is json ---------------

compress: # [string] Compress - Choose data compression format to apply before m

# -------------------------------------------------------


# -------------- if format is parquet ---------------

parquetSchema: # [string] Parquet schema - Select a Parquet schema. New schemas
parquetRowGroupSize: # [string] Row group size - Ideal memory size for row group
parquetPageSize: # [string] Page size - Ideal memory size for page segments. E.g
spacer: # [null]
parquetVersion: # [string] Parquet version - Determines which data types are sup
parquetDataPageVersion: # [string] Data page version - Serialization format of d
shouldLogInvalidRows: # [boolean] Log invalid rows - Output up to 20 unique rows

# -------------------------------------------------------

baseFileName: # [string] File name prefix expression - JavaScript expression to
fileNameSuffix: # [string] File name suffix expression - JavaScript expression t
maxFileSizeMB: # [number] Max file size (MB) - Maximum uncompressed output file
maxFileOpenTimeSec: # [number] Max file open time (Sec) - Maximum amount of time
maxFileIdleTimeSec: # [number] Max file idle time (Sec) - Maximum amount of time
maxOpenFiles: # [number] Max open files - Maximum number of files to keep open c
onBackpressure: # [string] Backpressure behavior - Whether to block or drop even
authType: # [string] Authentication method - Enter connection string directly, o
connectionString: # [string] Connection string - Enter your Azure Storage accoun

# -------------- if authType is manual ---------------


# -------------------------------------------------------

textSecret: # [string] Connection string (text secret) - Select or create a stor
```

```
  # -------------- if authType is secret ---------------


  # ------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to tl
  systemFields: # [array of strings] System fields - Set of fields to automatically
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in St
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] Request concurrency - Maximum number of ongoing requests
  maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the req
  maxPayloadEvents: # [number] Max events per request - Max number of events to inc
  compress: # [boolean] Compress - Whether to compress the payload body before send
  rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are not
  timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
  extraHttpHeaders: # [array] Extra HTTP headers - Extra HTTP headers.
    - name: # [string] Name - Field name
      value: # [string] Value - Field value
  useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS loo
  failedRequestLoggingMode: # [string] Failed request logging mode - Determines whi
  safeHeaders: # [array of strings] Safe headers - List of headers that are safe to
  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or que

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] Max file size - The maximum size to store in each queue
  pqMaxSize: # [string] Max queue size - The maximum amount of disk space the queue
  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 - Whether to block or drop event
  pqControls: # [object]

  # ------------------------------------------------------

  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 tha

  # -------------- if authType is secret ---------------

  # ----------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to t
  systemFields: # [array of strings] System fields - Set of fields to automatical
  environment: # [string] Environment - Optionally, enable this config only on a s
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in S
kinesis_output: # [object]
  type: # [string] Output Type
  streamName: # [string] Stream Name - Kinesis stream name to send events to.
  awsAuthenticationMethod: # [string] Authentication method - AWS authentication m

  # -------------- if awsAuthenticationMethod is manual ---------------

  awsApiKey: # [string] Access key - Access key

  # ----------------------------------------------------

  # -------------- if awsAuthenticationMethod is secret ---------------

  awsSecret: # [string] Secret key pair - Select or create a stored secret that re

  # ----------------------------------------------------

  awsSecretKey: # [string] Secret key - Secret key
  region: # [string] [required] Region - Region where the Kinesis stream is locate
  endpoint: # [string] Endpoint - Kinesis stream service endpoint. If empty, defau
  signatureVersion: # [string] Signature version - Signature version to use for si
  reuseConnections: # [boolean] Reuse connections - Whether to reuse connections b
  rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to re
  enableAssumeRole: # [boolean] Enable for Kinesis stream - Use Assume Role creden
  assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the rol
  assumeRoleExternalId: # [string] External ID - External ID to use when assuming
```

```
  assumeRoleExternalId: # [string] External ID - External ID to use when assuming
  concurrency: # [number] Put request concurrency - Maximum number of ongoing put
  maxRecordSizeKB: # [number] Max record size (KB, uncompressed) - Maximum size (KB
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sr
  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or que

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] Max file size - The maximum size to store in each queue
  pqMaxSize: # [string] Max queue size - The maximum amount of disk space the queue
  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 - Whether to block or drop event
  pqControls: # [object]

  # ----------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Set of fields to automatically
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in St
honeycomb_output: # [object]
  type: # [string] Output Type
  dataset: # [string] Dataset name - Name of the dataset to send events to â e.g
  concurrency: # [number] Request concurrency - Maximum number of ongoing requests
  maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the req
  maxPayloadEvents: # [number] Max events per request - Max number of events to inc
  compress: # [boolean] Compress - Whether to compress the payload body before send
  rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are not
  timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sr
  extraHttpHeaders: # [array] Extra HTTP headers - Extra HTTP headers.
    - name: # [string] Name - Field name
      value: # [string] Value - Field value
  useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS loo
  failedRequestLoggingMode: # [string] Failed request logging mode - Determines whi
  safeHeaders: # [array of strings] Safe headers - List of headers that are safe to
  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or que

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] Max file size - The maximum size to store in each queue
  pqMaxSize: # [string] Max queue size - The maximum amount of disk space the queue
  pqPath: # [string] Queue file path - The location for the persistent queue files
```

```
  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 - Whether to block or drop event
  pqControls: # [object]

  # ------------------------------------------------------

  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 se

  # -------------- if authType is secret ---------------


  # ------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to t
  systemFields: # [array of strings] System fields - Set of fields to automatically
  environment: # [string] Environment - Optionally, enable this config only on a s
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in S
azure_eventhub_output: # [object]
  type: # [string] Output Type
  brokers: # [array of strings] Brokers - List of Event Hubs Kafka brokers to conn
  topic: # [string] [required] Event Hub Name - The name of the Event Hub (a.k.a.
  ack: # [number] Acknowledgments - Control the number of required acknowledgments
  format: # [string] Record data format - Format to use to serialize events before
  maxRecordSizeKB: # [number] Max record size (KB, uncompressed) - Maximum size (K
  flushEventCount: # [number] Max events per batch - Maximum number of events in a
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. S
  connectionTimeout: # [number] Connection timeout (ms) - Maximum time to wait for
  requestTimeout: # [number] Request timeout (ms) - Maximum time to wait for a suc
  sasl: # [object] Authentication - Authentication parameters to use when connecti
    disabled: # [boolean] Disabled - Enable authentication.

    # -------------- if disabled is false ---------------

    mechanism: # [string] SASL mechanism - SASL authentication mechanism to use. P
    username: # [string] Username - The username for authentication. For Event Hub
```

```
    username: # [string] Username - The username for authentication. For Event Hubs
    authType: # [string] Authentication method - Enter password directly, or selec

    # ------------------------------------------------------

  tls: # [object] TLS settings (client side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    rejectUnauthorized: # [boolean] Validate server certs - For Event Hubs, this sh

    # ------------------------------------------------------

  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or que

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] Max file size - The maximum size to store in each queue
  pqMaxSize: # [string] Max queue size - The maximum amount of disk space the queue
  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 - Whether to block or drop event
  pqControls: # [object]

  # ------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Set of fields to automatically
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in St
google_chronicle_output: # [object]
  type: # [string] Output Type
  authenticationMethod: # [string] Authentication Method

  # -------------- if authenticationMethod is manual ---------------

  apiKey: # [string] API key - Organization's API key in Google Chronicle

  # ------------------------------------------------------


  # -------------- if authenticationMethod is secret ---------------
```

```
  apiKeySecret: # [string] API key (text secret) - Select or create a stored text s

  # ------------------------------------------------------

  logType: # [string] Log type - Log type value to send to Chronicle. Can be overwi
  logTextField: # [string] Log text field - Name of the event field that contains t
  logFormatType: # [string] Send events as
  region: # [string] Region - Regional endpoint to send events to
  concurrency: # [number] Request concurrency - Maximum number of ongoing requests
  maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the rec
  maxPayloadEvents: # [number] Max events per request - Max number of events to inc
  compress: # [boolean] Compress - Whether to compress the payload body before send
  rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are not
  timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
  extraHttpHeaders: # [array] Extra HTTP headers - Extra HTTP headers.
    - name: # [string] Name - Field name
      value: # [string] Value - Field value
  useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS loo
  failedRequestLoggingMode: # [string] Failed request logging mode - Determines whi
  safeHeaders: # [array of strings] Safe headers - List of headers that are safe to
  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or que

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] Max file size - The maximum size to store in each queue
  pqMaxSize: # [string] Max queue size - The maximum amount of disk space the queue
  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 - Whether to block or drop event
  pqControls: # [object]

  # ------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Set of fields to automatically
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in St
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
awsSecretKey: # [string] Secret - HMAC secret

# --------------------------------------------------------


# -------------- if awsAuthenticationMethod is secret ---------------

awsSecret: # [string] Secret key pair - Select or create a stored secret that re

# --------------------------------------------------------

stagePath: # [string] [required] Staging location - Filesystem location in which
destPath: # [string] [required] Key prefix - Prefix to append to files before up
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 - Whether to reuse connections b
rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to re
addIdToStagePath: # [boolean] Add output ID - Append output's ID to staging loca
removeEmptyDirs: # [boolean] Remove staging dirs - Remove empty staging director

# -------------- if removeEmptyDirs is true ---------------

emptyDirCleanupSec: # [number] Staging cleanup period - How often (secs) to clea

# --------------------------------------------------------

partitionExpr: # [string] Partitioning expression - JS expression defining how f
format: # [string] Data format - Format of the output data.

# -------------- if format is json ---------------

compress: # [string] Compress - Choose data compression format to apply before m

# --------------------------------------------------------
```

```
  # -------------- if format is parquet ---------------

  parquetSchema: # [string] Parquet schema - Select a Parquet schema. New schemas
  parquetRowGroupSize: # [string] Row group size - Ideal memory size for row group
  parquetPageSize: # [string] Page size - Ideal memory size for page segments. E.g
  spacer: # [null]
  parquetVersion: # [string] Parquet version - Determines which data types are supp
  parquetDataPageVersion: # [string] Data page version - Serialization format of da
  shouldLogInvalidRows: # [boolean] Log invalid rows - Output up to 20 unique rows

  # ------------------------------------------------------

  baseFileName: # [string] File name prefix expression - JavaScript expression to
  fileNameSuffix: # [string] File name suffix expression - JavaScript expression to
  maxFileSizeMB: # [number] Max file size (MB) - Maximum uncompressed output file s
  maxFileOpenTimeSec: # [number] Max file open time (Sec) - Maximum amount of time
  maxFileIdleTimeSec: # [number] Max file idle time (Sec) - Maximum amount of time
  maxOpenFiles: # [number] Max open files - Maximum number of files to keep open co
  onBackpressure: # [string] Backpressure behavior - Whether to block or drop event
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Set of fields to automatically
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in St
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
  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] Authentication Method - Google authentication method

  # -------------- 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] Batch size - The maximum number of items the Google API sh
  batchTimeout: # [number] Batch timeout (ms) - The maximum amount of time, in mil
  maxQueueSize: # [number] Max queue size - Maximum number of queued batches befor
  maxRecordSizeKB: # [number] Max batch size (KB) - Maximum size (KB) of batches t
  maxInProgress: # [number] Max concurrent requests - The maximum number of in-pro
  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or qu

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] Max file size - The maximum size to store in each queu
  pqMaxSize: # [string] Max queue size - The maximum amount of disk space the queu
  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 - Whether to block or drop even
  pqControls: # [object]

  # ------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to t
  systemFields: # [array of strings] System fields - Set of fields to automatical
  environment: # [string] Environment - Optionally, enable this config only on a s
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in S
kafka_output: # [object]
  type: # [string] Output Type
  brokers: # [array of strings] Brokers - List of Kafka brokers to connect to, e.g
  topic: # [string] [required] Topic - The topic to publish events to. Can be over
  ack: # [number] Acknowledgments - Control the number of required acknowledgments
  format: # [string] Record data format - Format to use to serialize events before
  compression: # [string] Compression - Codec to use to compress the data before s
  maxRecordSizeKB: # [number] Max record size (KB, uncompressed) - Maximum size (K
  flushEventCount: # [number] Max events per batch - Maximum number of events in a
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. S
  kafkaSchemaRegistry: # [object] Kafka Schema Registry Authentication
    disabled: # [boolean] Disabled - Enable Schema Registry

    # -------------- if disabled is false ---------------

    schemaRegistryURL: # [string] Schema Registry URL - URL for access to the Conf
    defaultKeySchemaId: # [number] Default key schema ID - Used when __keySchemaId
    defaultValueSchemaId: # [number] Default value schema ID - Used when __valueSc
    tls: # [object] TLS settings (client side)
      disabled: # [boolean] Disabled
```

```
    # -------------- if disabled is false ---------------

    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that ar
    servername: # [string] Server name (SNI) - Server name for the SNI (Server Na
    certificateName: # [string] Certificate name - The name of the predefined ce
    caPath: # [string] CA certificate path - Path on client in which to find CA
    privKeyPath: # [string] Private key path (mutual auth) - Path on client in wh
    certPath: # [string] Certificate path (mutual auth) - Path on client in whic
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
    minVersion: # [string] Minimum TLS version - Minimum TLS version to use when
    maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when

    # ------------------------------------------------------


  # --------------------------------------------------------

connectionTimeout: # [number] Connection timeout (ms) - Maximum time to wait for
requestTimeout: # [number] Request timeout (ms) - Maximum time to wait for a suc
sasl: # [object] Authentication - Authentication parameters to use when connecti
  disabled: # [boolean] Disabled - Enable Authentication

  # -------------- if disabled is false ---------------

  mechanism: # [string] SASL mechanism - SASL authentication mechanism to use.

  # ---------------------------------------------------------

tls: # [object] TLS settings (client side)
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
  servername: # [string] Server name (SNI) - Server name for the SNI (Server Nam
  certificateName: # [string] Certificate name - The name of the predefined cert
  caPath: # [string] CA certificate path - Path on client in which to find CA ce
  privKeyPath: # [string] Private key path (mutual auth) - Path on client in whi
  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 - Minimum TLS version to use when c
  maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when c
```

```
    # ------------------------------------------------------

  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or qu

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] Max file size - The maximum size to store in each queu
  pqMaxSize: # [string] Max queue size - The maximum amount of disk space the queu
  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 - Whether to block or drop even
  pqControls: # [object]

  # -------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to t
  systemFields: # [array of strings] System fields - Set of fields to automatical
  environment: # [string] Environment - Optionally, enable this config only on a s
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in S
confluent_cloud_output: # [object]
  type: # [string] Output Type
  brokers: # [array of strings] Brokers - List of Confluent Cloud brokers to conne
  tls: # [object] TLS settings (client side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
    servername: # [string] Server name (SNI) - Server name for the SNI (Server Nam
    certificateName: # [string] Certificate name - The name of the predefined cert
    caPath: # [string] CA certificate path - Path on client in which to find CA ce
    privKeyPath: # [string] Private key path (mutual auth) - Path on client in whi
    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 - Minimum TLS version to use when c
    maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when c

    # -------------------------------------------------------

  topic: # [string] [required] Topic - The topic to publish events to. Can be over
  ack: # [number] Acknowledgments - Control the number of required acknowledgments
  format: # [string] Record data format - Format to use to serialize events before
```

```
compression: # [string] Compression - Codec to use to compress the data before s
maxRecordSizeKB: # [number] Max record size (KB, uncompressed) - Maximum size (KI
flushEventCount: # [number] Max events per batch - Maximum number of events in a
flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sr
kafkaSchemaRegistry: # [object] Kafka Schema Registry Authentication
  disabled: # [boolean] Disabled - Enable Schema Registry

  # -------------- if disabled is false ---------------

  schemaRegistryURL: # [string] Schema Registry URL - URL for access to the Confl
  defaultKeySchemaId: # [number] Default key schema ID - Used when __keySchemaIdC
  defaultValueSchemaId: # [number] Default value schema ID - Used when __valueScl
  tls: # [object] TLS settings (client side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
    servername: # [string] Server name (SNI) - Server name for the SNI (Server Na
    certificateName: # [string] Certificate name - The name of the predefined cer
    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 wl
    certPath: # [string] Certificate path (mutual auth) - Path on client in whicl
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
    minVersion: # [string] Minimum TLS version - Minimum TLS version to use when
    maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when

    # ----------------------------------------------------------


  # ----------------------------------------------------------

connectionTimeout: # [number] Connection timeout (ms) - Maximum time to wait for
requestTimeout: # [number] Request timeout (ms) - Maximum time to wait for a succ
sasl: # [object] Authentication - Authentication parameters to use when connecti
  disabled: # [boolean] Disabled - Enable Authentication

  # -------------- if disabled is false ---------------

  mechanism: # [string] SASL mechanism - SASL authentication mechanism to use.

  # ----------------------------------------------------------
```

```
  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or qu

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] Max file size - The maximum size to store in each queu
  pqMaxSize: # [string] Max queue size - The maximum amount of disk space the queu
  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 - Whether to block or drop even
  pqControls: # [object]

  # ---------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to t
  systemFields: # [array of strings] System fields - Set of fields to automatical
  environment: # [string] Environment - Optionally, enable this config only on a s
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in S
elastic_output: # [object]
  type: # [string] Output Type
  loadBalanced: # [boolean] Load balancing - Use load-balanced destinations

  # -------------- if loadBalanced is false ---------------

  url: # [string] Bulk API URL or Cloud ID - Enter Cloud ID or URL to an Elastic c
  useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS lo

  # ---------------------------------------------------------


  # -------------- if loadBalanced is true ---------------

  excludeSelf: # [boolean] Exclude current host IPs - Exclude all IPs of the curre
  urls: # [array] Bulk API URLs
    - url: # [string] URL - URL to an Elastic node to send events to â e.g., htt
      weight: # [number] Load Weight - The weight to use for load-balancing purpos
  dnsResolvePeriodSec: # [number] DNS resolution period (seconds) - Re-resolve any
  loadBalanceStatsPeriodSec: # [number] Load balance stats period (seconds) - How

  # ---------------------------------------------------------

  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 b
  concurrency: # [number] Request concurrency - Maximum number of ongoing requests
```

```
  maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the re
  maxPayloadEvents: # [number] Max events per request - Max number of events to in
  compress: # [boolean] Compress - Whether to compress the payload body before sen
  rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are no
  timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. S
  extraHttpHeaders: # [array] Extra HTTP headers - Extra HTTP headers.
    - name: # [string] Name - Field name
      value: # [string] Value - Field value
  failedRequestLoggingMode: # [string] Failed request logging mode - Determines wh
  safeHeaders: # [array of strings] Safe headers - List of headers that are safe t
  extraParams: # [array] Extra Parameters - Extra Parameters.
    - name: # [string] Name - Field name
      value: # [string] Value - Field value
  auth: # [object]
    disabled: # [boolean] Authentication Disabled

    # -------------- if disabled is false ---------------

    authType: # [string] Authentication method - Enter credentials directly, or se

    # ---------------------------------------------------

  elasticVersion: # [string] Elastic Version - Optional Elasticsearch version, use
  elasticPipeline: # [string] Elastic pipeline - Optional Elasticsearch destinatio
  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or qu

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] Max file size - The maximum size to store in each queu
  pqMaxSize: # [string] Max queue size - The maximum amount of disk space the queu
  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 - Whether to block or drop even
  pqControls: # [object]

  # ---------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to t
  systemFields: # [array of strings] System fields - Set of fields to automatical
  environment: # [string] Environment - Optionally, enable this config only on a s
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in S
newrelic_output: # [object]
```

```
type: # [string] Output Type
region: # [string] Region - Which New Relic region endpoint to use.
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 messag
metadata: # [array] Fields - Fields to add to events from this input.
  - name: # [string] Name - Field name
    value: # [string] Value - JavaScript expression to compute field's value, en
concurrency: # [number] Request concurrency - Maximum number of ongoing requests
maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the re
maxPayloadEvents: # [number] Max events per request - Max number of events to in
compress: # [boolean] Compress - Whether to compress the payload body before sen
rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are no
timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. S
extraHttpHeaders: # [array] Extra HTTP headers - Extra HTTP headers.
  - name: # [string] Name - Field name
    value: # [string] Value - Field value
useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS lo
failedRequestLoggingMode: # [string] Failed request logging mode - Determines wh
safeHeaders: # [array of strings] Safe headers - List of headers that are safe t
onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or qu

# -------------- if onBackpressure is queue ---------------

pqMaxFileSize: # [string] Max file size - The maximum size to store in each queu
pqMaxSize: # [string] Max queue size - The maximum amount of disk space the queu
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 - Whether to block or drop even
pqControls: # [object]

# --------------------------------------------------------

authType: # [string] Authentication method - Enter API key directly, or select a
apiKey: # [string] API key - New Relic API key. Can be overridden using __newRel

# -------------- if authType is manual ---------------


# --------------------------------------------------------

textSecret: # [string] API key (text secret) - Select or create a stored text se
```

```
  # -------------- if authType is secret ---------------


  # -----------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to tl
  systemFields: # [array of strings] System fields - Set of fields to automatically
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in St
newrelic_events_output: # [object]
  type: # [string] Output Type
  region: # [string] [required] Region - Which New Relic region endpoint to use.
  accountId: # [string] Account ID - New Relic account ID
  eventType: # [string] [required] Event type - Default eventType to use when not p
  concurrency: # [number] Request concurrency - Maximum number of ongoing requests
  maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the req
  maxPayloadEvents: # [number] Max events per request - Max number of events to inc
  compress: # [boolean] Compress - Whether to compress the payload body before send
  rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are not
  timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
  extraHttpHeaders: # [array] Extra HTTP headers - Extra HTTP headers.
    - name: # [string] Name - Field name
      value: # [string] Value - Field value
  useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS loo
  failedRequestLoggingMode: # [string] Failed request logging mode - Determines whi
  safeHeaders: # [array of strings] Safe headers - List of headers that are safe to
  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or que

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] Max file size - The maximum size to store in each queue
  pqMaxSize: # [string] Max queue size - The maximum amount of disk space the queue
  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 - Whether to block or drop event
  pqControls: # [object]

  # -----------------------------------------------------

  authType: # [string] Authentication method - Enter API key directly, or select a
  apiKey: # [string] API key - New Relic API key. Can be overridden using __newRel
```

```
  # -------------- if authType is manual ---------------


  # ------------------------------------------------------

  textSecret: # [string] API key (text secret) - Select or create a stored text se

  # -------------- if authType is secret ---------------


  # ------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to t
  systemFields: # [array of strings] System fields - Set of fields to automatical
  environment: # [string] Environment - Optionally, enable this config only on a s
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in S
influxdb_output: # [object]
  type: # [string] Output Type
  url: # [string] Write API URL - URL of an InfluxDB cluster to send events to, e.
  useV2API: # [boolean] Use v2 API - The v2 API can be enabled with InfluxDB versi

  # -------------- 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
  dynamicValueFieldName: # [boolean] Dynamic value fields - Enabling this will pul
  valueFieldName: # [string] Value field name - Name of the field in which to stor
  concurrency: # [number] Request concurrency - Maximum number of ongoing requests
  maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the re
  maxPayloadEvents: # [number] Max events per request - Max number of events to in
  compress: # [boolean] Compress - Whether to compress the payload body before sen
  rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are no
```

```
timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
extraHttpHeaders: # [array] Extra HTTP headers - Extra HTTP headers.
  - name: # [string] Name - Field name
    value: # [string] Value - Field value
useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS loo
failedRequestLoggingMode: # [string] Failed request logging mode - Determines whi
safeHeaders: # [array of strings] Safe headers - List of headers that are safe to
onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or que

# -------------- if onBackpressure is queue ---------------

pqMaxFileSize: # [string] Max file size - The maximum size to store in each queue
pqMaxSize: # [string] Max queue size - The maximum amount of disk space the queue
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 - Whether to block or drop event
pqControls: # [object]

# ---------------------------------------------------------

authType: # [string] Authentication type - InfluxDB authentication type

# -------------- if authType is basic ---------------

username: # [string] Username - Username for Basic authentication
password: # [string] Password - Password for Basic authentication

# ---------------------------------------------------------


# -------------- 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 secr

  # ---------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to t
  systemFields: # [array of strings] System fields - Set of fields to automatical
  environment: # [string] Environment - Optionally, enable this config only on a s
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in S
cloudwatch_output: # [object]
  type: # [string] Output Type
  logGroupName: # [string] Log group name - CloudWatch log group to associate even
  logStreamName: # [string] [required] Log stream prefix - Prefix for CloudWatch l
  awsAuthenticationMethod: # [string] Authentication method - AWS authentication m

  # -------------- if awsAuthenticationMethod is manual ---------------

  awsApiKey: # [string] Access key - Access key

  # ---------------------------------------------------------


  # -------------- if awsAuthenticationMethod is secret ---------------

  awsSecret: # [string] Secret key pair - Select or create a stored secret that re

  # ---------------------------------------------------------

  awsSecretKey: # [string] Secret key - Secret key
  region: # [string] [required] Region - Region where the CloudWatchLogs is locate
  endpoint: # [string] Endpoint - CloudWatchLogs service endpoint. If empty, defau
  signatureVersion: # [string] Signature version - Signature version to use for si
  reuseConnections: # [boolean] Reuse connections - Whether to reuse connections b
  rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to re
  enableAssumeRole: # [boolean] Enable for CloudWatchLogs - Use Assume Role creden
  assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the rol
  assumeRoleExternalId: # [string] External ID - External ID to use when assuming
  maxQueueSize: # [number] Max queue size - Maximum number of queued batches befor
  maxRecordSizeKB: # [number] Max record size (KB, uncompressed) - Maximum size (K
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. S
```

```
  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or qu

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] Max file size - The maximum size to store in each queu
  pqMaxSize: # [string] Max queue size - The maximum amount of disk space the queu
  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 - Whether to block or drop even
  pqControls: # [object]

  # ------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to t
  systemFields: # [array of strings] System fields - Set of fields to automatical
  environment: # [string] Environment - Optionally, enable this config only on a s
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in S
minio_output: # [object]
  type: # [string] Output Type
  endpoint: # [string] [required] MinIO endpoint - MinIO service url (e.g. http://
  bucket: # [string] MinIO bucket name - Name of the destination MinIO bucket. Thi
  awsAuthenticationMethod: # [string] Authentication method - AWS authentication m

  # -------------- if awsAuthenticationMethod is manual ---------------

  awsApiKey: # [string] Access key - Access key

  # ------------------------------------------------------



  # -------------- if awsAuthenticationMethod is secret ---------------

  awsSecret: # [string] Secret key pair - Select or create a stored secret that re

  # ------------------------------------------------------

  awsSecretKey: # [string] Secret key - Secret key
  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 - Append output's ID to staging loca
  removeEmptyDirs: # [boolean] Remove staging dirs - Remove empty staging director

  # -------------- if removeEmptyDirs is true ---------------
```

```
# removeEmptyDirs is true

emptyDirCleanupSec: # [number] Staging cleanup period - How often (secs) to clear

# ------------------------------------------------------

destPath: # [string] [required] Key prefix - Root directory to prepend to path be
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 - Whether to reuse connections be
rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to re
partitionExpr: # [string] Partitioning expression - JS expression defining how fi
format: # [string] Data format - Format of the output data.

# -------------- if format is json ---------------

compress: # [string] Compress - Choose data compression format to apply before mo

# ------------------------------------------------------


# -------------- if format is parquet ---------------

parquetSchema: # [string] Parquet schema - Select a Parquet schema. New schemas c
parquetRowGroupSize: # [string] Row group size - Ideal memory size for row group
parquetPageSize: # [string] Page size - Ideal memory size for page segments. E.g
spacer: # [null]
parquetVersion: # [string] Parquet version - Determines which data types are supp
parquetDataPageVersion: # [string] Data page version - Serialization format of da
shouldLogInvalidRows: # [boolean] Log invalid rows - Output up to 20 unique rows

# ------------------------------------------------------

baseFileName: # [string] File name prefix expression - JavaScript expression to c
fileNameSuffix: # [string] File name suffix expression - JavaScript expression to
maxFileSizeMB: # [number] Max file size (MB) - Maximum uncompressed output file s
maxFileOpenTimeSec: # [number] Max file open time (Sec) - Maximum amount of time
maxFileIdleTimeSec: # [number] Max file idle time (Sec) - Maximum amount of time
maxOpenFiles: # [number] Max open files - Maximum number of files to keep open co
onBackpressure: # [string] Backpressure behavior - Whether to block or drop event
pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
systemFields: # [array of strings] System fields - Set of fields to automatically
```

```
  systemFields: # [array of strings] System fields - Set of fields to automaticall
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in St
statsd_output: # [object]
  type: # [string] Output Type
  protocol: # [string] Destination Protocol - Protocol to use when communicating wi

  # -------------- if protocol is tcp ---------------

  throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to thrott
  connectionTimeout: # [number] Connection Timeout - Amount of time (milliseconds)
  writeTimeout: # [number] Write Timeout - Amount of time (milliseconds) to wait fo
  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or que

  # -------------------------------------------------------

  host: # [string] [required] Host - The hostname of the destination.
  port: # [number] [required] Port - Destination port.
  mtu: # [number] Max record Size (Bytes) - Used when Protocol is UDP, to specify t
  flushPeriodSec: # [number] Flush period (sec) - Used when Protocol is TCP, to spe
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Set of fields to automatically
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in St
statsd_ext_output: # [object]
  type: # [string] Output Type
  protocol: # [string] Destination Protocol - Protocol to use when communicating wi

  # -------------- if protocol is tcp ---------------

  throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to thrott
  connectionTimeout: # [number] Connection Timeout - Amount of time (milliseconds)
  writeTimeout: # [number] Write Timeout - Amount of time (milliseconds) to wait fo
  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or que

  # -------------------------------------------------------

  host: # [string] [required] Host - The hostname of the destination.
  port: # [number] [required] Port - Destination port.
  mtu: # [number] Max record Size (Bytes) - Used when Protocol is UDP, to specify t
  flushPeriodSec: # [number] Flush period (sec) - Used when Protocol is TCP, to spe
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Set of fields to automatically
  environment: # [string] Environment - Optionally, enable this config only on a sp
```

```
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in St
graphite_output: # [object]
  type: # [string] Output Type
  protocol: # [string] Destination Protocol - Protocol to use when communicating wi

  # -------------- if protocol is tcp ---------------

  throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to thrott
  connectionTimeout: # [number] Connection Timeout - Amount of time (milliseconds)
  writeTimeout: # [number] Write Timeout - Amount of time (milliseconds) to wait fo
  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or que

  # -------------------------------------------------------

  host: # [string] [required] Host - The hostname of the destination.
  port: # [number] [required] Port - Destination port.
  mtu: # [number] Max record Size (Bytes) - Used when Protocol is UDP, to specify t
  flushPeriodSec: # [number] Flush period (sec) - Used when Protocol is TCP, to spe
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Set of fields to automatically
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in St
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 be
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Set of fields to automatically
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in St
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 ---------------
```

```
  # -------------- if awsAuthenticationMethod is manual ---------------

  awsApiKey: # [string] Access key - Access key

  # ------------------------------------------------------


  # -------------- if awsAuthenticationMethod is secret ---------------

  awsSecret: # [string] Secret key pair - Select or create a stored secret that re

  # ------------------------------------------------------

  awsSecretKey: # [string] Secret key - Secret key
  region: # [string] Region - AWS Region where the SQS queue is located. Required,
  endpoint: # [string] Endpoint - SQS service endpoint. If empty, defaults to AWS'
  signatureVersion: # [string] Signature version - Signature version to use for sig
  reuseConnections: # [boolean] Reuse connections - Whether to reuse connections be
  rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to re
  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
  maxQueueSize: # [number] Max queue size - Maximum number of queued batches before
  maxRecordSizeKB: # [number] Max record size (KB) - Maximum size (KB) of batches
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
  maxInProgress: # [number] Max concurrent requests - The maximum number of in-prog
  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or que

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] Max file size - The maximum size to store in each queue
  pqMaxSize: # [string] Max queue size - The maximum amount of disk space the queue
  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 - Whether to block or drop event
  pqControls: # [object]

  # ------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Set of fields to automatically
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in St
snmp_output: # [object]
```

```
snmp_output: # [object]
  type: # [string] Output Type
  hosts: # [array] SNMP Trap Destinations - One or more SNMP destinations to forwa
    - host: # [string] Address - Destination host
      port: # [number] Port - Destination port, default is 162
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Set of fields to automatically
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in St
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 - Optionally, override the source nam
  customCategory: # [string] Custom source category - Optionally, override the sour
  concurrency: # [number] Request concurrency - Maximum number of ongoing requests
  maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the req
  maxPayloadEvents: # [number] Max events per request - Max number of events to inc
  compress: # [boolean] Compress - Whether to compress the payload body before send
  rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are not
  timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
  extraHttpHeaders: # [array] Extra HTTP headers - Extra HTTP headers.
    - name: # [string] Name - Field name
      value: # [string] Value - Field value
  useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS loo
  failedRequestLoggingMode: # [string] Failed request logging mode - Determines whi
  safeHeaders: # [array of strings] Safe headers - List of headers that are safe to
  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or que

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] Max file size - The maximum size to store in each queue
  pqMaxSize: # [string] Max queue size - The maximum amount of disk space the queue
  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 - Whether to block or drop event
  pqControls: # [object]

  # ---------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Set of fields to automatically
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in St
```

```
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in S
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
  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 sen
  tags: # [array of strings] Datadog tags - List of tags to send with logs (e.g.,
  allowApiKeyFromEvents: # [boolean] Allow API key from events - If enabled, the AP
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in St
  severity: # [string] Severity - Default value for message severity. When you send
  site: # [string] Datadog site
  concurrency: # [number] Request concurrency - Maximum number of ongoing requests
  maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the req
  maxPayloadEvents: # [number] Max events per request - Max number of events to inc
  compress: # [boolean] Compress - Whether to compress the payload body before send
  rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are not
  timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
  extraHttpHeaders: # [array] Extra HTTP headers - Extra HTTP headers.
    - name: # [string] Name - Field name
      value: # [string] Value - Field value
  useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS loo
  failedRequestLoggingMode: # [string] Failed request logging mode - Determines whi
  safeHeaders: # [array of strings] Safe headers - List of headers that are safe to
  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or que

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] Max file size - The maximum size to store in each queue
  pqMaxSize: # [string] Max queue size - The maximum amount of disk space the queue
  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 - Whether to block or drop event
  pqControls: # [object]

  # ---------------------------------------------------------

  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 se

  # -------------- if authType is secret ---------------


  # ------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to t
  systemFields: # [array of strings] System fields - Set of fields to automatical
  environment: # [string] Environment - Optionally, enable this config only on a s
grafana_cloud_output: # [object]
  type: # [string] Output Type
  lokiUrl: # [string] Loki URL - The endpoint to send logs to, e.g.: https://logs-
  prometheusUrl: # [string] [required] Prometheus URL - The remote_write endpoint
  message: # [string] Logs message field - Name of the event field that contains t
  messageFormat: # [string] Message Format - Which format to use when sending logs

  # -------------- if messageFormat is json ---------------

  compress: # [boolean] Compress - Whether to compress the payload body before sen

  # ------------------------------------------------------

  labels: # [array] Logs labels - List of labels to send with logs. Labels define
    - name: # [string] Name - Name of the label.
      value: # [string] Value - Value of the label.
  metricRenameExpr: # [string] Metrics renaming expression - A JS expression that
  prometheusAuth: # [object]
    authType: # [string] Authentication Type - The authentication method to use fo

    # -------------- if authType is token ---------------

    token: # [string] Auth token - Bearer token to include in the authorization he

    # ------------------------------------------------------


    # -------------- if authType is textSecret ---------------
```

```
    textSecret: # [string] Auth token (text secret) - Select or create a stored te

    # ---------------------------------------------------------


    # -------------- if authType is basic ---------------

    username: # [string] Username - Username for authentication
    password: # [string] Password - Password (a.k.a API key in Grafana Cloud domai

    # ---------------------------------------------------------


    # -------------- if authType is credentialsSecret ---------------

    credentialsSecret: # [string] Credentials secret - Select or create a secret t

    # ---------------------------------------------------------

  lokiAuth: # [object]
    authType: # [string] Authentication Type - The authentication method to use fo

    # -------------- if authType is token ---------------

    token: # [string] Auth token - Bearer token to include in the authorization he

    # ---------------------------------------------------------


    # -------------- if authType is textSecret ---------------

    textSecret: # [string] Auth token (text secret) - Select or create a stored te

    # ---------------------------------------------------------


    # -------------- if authType is basic ---------------

    username: # [string] Username - Username for authentication
    password: # [string] Password - Password (a.k.a API key in Grafana Cloud domai

    # ---------------------------------------------------------
```

```
    # -------------- if authType is credentialsSecret ---------------

    credentialsSecret: # [string] Credentials secret - Select or create a secret th

    # ----------------------------------------------------

  concurrency: # [number] Request concurrency - Maximum number of ongoing requests
  maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the req
  maxPayloadEvents: # [number] Max events per request - Maximum number of events to
  rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are not
  timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sm
  extraHttpHeaders: # [array] Extra HTTP headers - Extra HTTP headers.
    - name: # [string] Name - Field name
      value: # [string] Value - Field value
  useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS loo
  failedRequestLoggingMode: # [string] Failed request logging mode - Determines whi
  safeHeaders: # [array of strings] Safe headers - List of headers that are safe to
  systemFields: # [array of strings] System fields - Set of fields to automatically
  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or que

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] Max file size - The maximum size to store in each queue
  pqMaxSize: # [string] Max queue size - The maximum amount of disk space the queue
  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 - Whether to block or drop event
  pqControls: # [object]

  # ----------------------------------------------------

  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 - Add tags for filtering and grouping in Sp
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 - Which format to use when sending logs

  # -------------- if messageFormat is json ---------------
```

```
compress: # [boolean] Compress - Whether to compress the payload body before sen

# -------------------------------------------------------

labels: # [array] Logs labels - List of labels to send with logs. Labels define
  - name: # [string] Name - Name of the label.
    value: # [string] Value - Value of the label.
authType: # [string] Authentication Type - The authentication method to use for

# -------------- if authType is token ---------------

token: # [string] Auth token - Bearer token to include in the authorization head

# -------------------------------------------------------


# -------------- 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 (a.k.a API key in Grafana Cloud domain)

# -------------------------------------------------------


# -------------- if authType is credentialsSecret ---------------

credentialsSecret: # [string] Credentials secret - Select or create a secret tha

# -------------------------------------------------------

concurrency: # [number] Request concurrency - Maximum number of ongoing requests
maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the re
maxPayloadEvents: # [number] Max events per request - Maximum number of events t
rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are no
timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
```

```
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sr
  extraHttpHeaders: # [array] Extra HTTP headers - Extra HTTP headers.
    - name: # [string] Name - Field name
      value: # [string] Value - Field value
  useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS loc
  failedRequestLoggingMode: # [string] Failed request logging mode - Determines wh
  safeHeaders: # [array of strings] Safe headers - List of headers that are safe t
  systemFields: # [array of strings] System fields - Set of fields to automatically
  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or qu

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] Max file size - The maximum size to store in each queu
  pqMaxSize: # [string] Max queue size - The maximum amount of disk space the queu
  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 - Whether to block or drop even
  pqControls: # [object]

  # ---------------------------------------------------------

  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 - Add tags for filtering and grouping in St
prometheus_output: # [object]
  type: # [string] Output Type
  url: # [string] Remote Write URL - The endpoint to send metrics to.
  metricRenameExpr: # [string] Metric renaming expression - A JS expression that ca
  sendMetadata: # [boolean] Send metadata - Whether to generate and send metadata (

  # -------------- if sendMetadata is true ---------------

  metricsFlushPeriodSec: # [number] Metadata flush period (sec) - How frequently me

  # ---------------------------------------------------------

  systemFields: # [array of strings] System fields - Set of fields to automatically
  concurrency: # [number] Request concurrency - Maximum number of ongoing requests
  maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the req
  maxPayloadEvents: # [number] Max events per request - Max number of events to inc
  rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are not
  timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Sr
```

```
extraHttpHeaders: # [array] Extra HTTP headers - Extra HTTP headers.
  - name: # [string] Name - Field name
    value: # [string] Value - Field value
useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS lo
failedRequestLoggingMode: # [string] Failed request logging mode - Determines wh
safeHeaders: # [array of strings] Safe headers - List of headers that are safe t
onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or qu

# -------------- if onBackpressure is queue ---------------

pqMaxFileSize: # [string] Max file size - The maximum size to store in each queu
pqMaxSize: # [string] Max queue size - The maximum amount of disk space the queu
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 - Whether to block or drop even
pqControls: # [object]

# ----------------------------------------------------------

authType: # [string] Authentication type - Remote Write authentication type

# -------------- if authType is basic ---------------

username: # [string] Username - Username for Basic authentication
password: # [string] Password - Password for Basic authentication

# ----------------------------------------------------------


# -------------- 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 tha

# ----------------------------------------------------------
```

```
  # -------------- if authType is textSecret ---------------

  textSecret: # [string] Token (text secret) - Select or create a stored text secr

  # ---------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to t
  environment: # [string] Environment - Optionally, enable this config only on a s
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in S
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
  maxDataSize: # [string] Max data size - Maximum disk space allowed to be consume
  maxDataTime: # [string] Max data age - Maximum amount of time to retain data (e.g
  compress: # [string] Compression - Select data compression format. Optional.
  destPath: # [string] Path location - Path to use to write metrics. Defaults to $C
  onBackpressure: # [string] Backpressure behavior - Whether to block or drop event
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in S
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to t
  systemFields: # [array of strings] System fields - Set of fields to automatically
  environment: # [string] Environment - Optionally, enable this config only on a s
open_telemetry_output: # [object]
  type: # [string] Output Type
  endpoint: # [string] Endpoint - The endpoint to send OTEL events to. It can be a
  authType: # [string] Authentication type - OpenTelemetry authentication type

  # -------------- if authType is basic ---------------

  username: # [string] Username - Username for Basic authentication
  password: # [string] Password - Password for Basic authentication

  # ---------------------------------------------------------


  # -------------- 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 tha

# ------------------------------------------------------


# -------------- if authType is textSecret ---------------

textSecret: # [string] Token (text secret) - Select or create a stored text secr

# ------------------------------------------------------

tls: # [object] TLS settings (client side)
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
  certificateName: # [string] Certificate name - The name of the predefined cert
  caPath: # [string] CA certificate path - Path on client in which to find CA ce
  privKeyPath: # [string] Private key path (mutual auth) - Path on client in whi
  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 - Minimum TLS version to use when c
  maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when c

  # -------------------------------------------------------

metadata: # [array] Metadata - Extra information to send with each gRPC request
  - key: # [string] Key - The key of the metadata.
    value: # [string] Value - The value of the metadata.
concurrency: # [number] Request concurrency - Maximum number of ongoing requests
maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the re
timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. S
failedRequestLoggingMode: # [string] Failed request logging mode - Determines wh
safeHeaders: # [array of strings] Safe headers - List of headers that are safe t
connectionTimeout: # [number] Connection Timeout - Amount of time (milliseconds)
keepAliveTime: # [number] Keep Alive Time (seconds) - How often the sender shoul
onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or qu

# -------------- if onBackpressure is queue ---------------
```

```
  pqMaxFileSize: # [string] Max file size - The maximum size to store in each queu
  pqMaxSize: # [string] Max queue size - The maximum amount of disk space the queu
  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 - Whether to block or drop even
  pqControls: # [object]

  # ------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to t
  systemFields: # [array of strings] System fields - Set of fields to automatically
  environment: # [string] Environment - Optionally, enable this config only on a s
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in S
dataset_output: # [object]
  type: # [string] Output Type
  messageField: # [string] Message field - Name of the event field that contains t
  excludeFields: # [array of strings] Exclude fields - Fields to exclude from the
  serverHostField: # [string] Server/host field - Name of the event field that con
  timestampField: # [string] Timestamp field - Name of the event field that contai
  defaultSeverity: # [string] Severity - Default value for event severity. If the
  site: # [string] DataSet site - DataSet site to which events should be sent
  customUrl: # [string]
  concurrency: # [number] Request concurrency - Maximum number of ongoing requests
  maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the re
  maxPayloadEvents: # [number] Max events per request - Max number of events to in
  compress: # [boolean] Compress - Whether to compress the payload body before sen
  rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are no
  timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. S
  extraHttpHeaders: # [array] Extra HTTP headers - Extra HTTP headers.
    - name: # [string] Name - Field name
      value: # [string] Value - Field value
  useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS lo
  failedRequestLoggingMode: # [string] Failed request logging mode - Determines wh
  safeHeaders: # [array of strings] Safe headers - List of headers that are safe t
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in S
  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or qu

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] Max file size - The maximum size to store in each queu
  pqMaxSize: # [string] Max queue size - The maximum amount of disk space the queu
  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 - Whether to block or drop even
  pqControls: # [object]

  # ------------------------------------------------------

  authType: # [string] Authentication method - Enter API key directly, or select a
  apiKey: # [string] API key - A 'Log Write Access' API key for the DataSet accoun

  # -------------- if authType is manual ---------------


  # ------------------------------------------------------

  textSecret: # [string] API key (text secret) - Select or create a stored text se

  # -------------- if authType is secret ---------------


  # ------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to t
  systemFields: # [array of strings] System fields - Set of fields to automatical
  environment: # [string] Environment - Optionally, enable this config only on a s
logstream_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] Port - The port to connect to on the provided host

  # ------------------------------------------------------


  # -------------- if loadBalanced is true ---------------

  excludeSelf: # [boolean] Exclude current host IPs - Exclude all IPs of the curre
  hosts: # [array] Destinations - Set of hosts to load-balance data to.
    - host: # [string] Address - The hostname of the receiver.
      port: # [number] Port - The port to connect to on the provided host.
      tls: # [string] TLS - Whether to inherit TLS configs from group setting or d
```

```
    servername: # [string] TLS Servername - Servername to use if establishing a
    weight: # [number] Load Weight - The weight to use for load-balancing purpos
dnsResolvePeriodSec: # [number] DNS resolution period (seconds) - Re-resolve any
loadBalanceStatsPeriodSec: # [number] Load balance stats period (seconds) - How
maxConcurrentSenders: # [number] Max connections - Maximum number of concurrent

# ------------------------------------------------------

compression: # [string] Compression - Codec to use to compress the data before s
throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to thrott
tls: # [object] TLS settings (client side)
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
  servername: # [string] Server name (SNI) - Server name for the SNI (Server Nam
  certificateName: # [string] Certificate name - The name of the predefined cert
  caPath: # [string] CA certificate path - Path on client in which to find CA ce
  privKeyPath: # [string] Private key path (mutual auth) - Path on client in whi
  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 - Minimum TLS version to use when c
  maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when c

  # ------------------------------------------------------

connectionTimeout: # [number] Connection Timeout - Amount of time (milliseconds)
writeTimeout: # [number] Write Timeout - Amount of time (milliseconds) to wait f
tokenTTLMinutes: # [number] Auth Token TTL minutes - The number of minutes befor
onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or qu

# -------------- if onBackpressure is queue ---------------

pqMaxFileSize: # [string] Max file size - The maximum size to store in each queu
pqMaxSize: # [string] Max queue size - The maximum amount of disk space the queu
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 - Whether to block or drop even
pqControls: # [object]

# ------------------------------------------------------
```

```
  authType: # [string] Authentication method - Enter a token directly, or provide a
  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 ---------------


  # -----------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to th
  systemFields: # [array of strings] System fields - Set of fields to automatically
  environment: # [string] Environment - Optionally, enable this config only on a sp
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in St
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] Port - The port to connect to on the provided host

  # -----------------------------------------------------


  # -------------- 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] Port - The port to connect to on the provided host.
      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 - The weight to use for load-balancing purpose
  dnsResolvePeriodSec: # [number] DNS resolution period (seconds) - Re-resolve any
  loadBalanceStatsPeriodSec: # [number] Load balance stats period (seconds) - How f
  maxConcurrentSenders: # [number] Max connections - Maximum number of concurrent c
```

```
# ------------------------------------------------------

compression: # [string] Compression - Codec to use to compress the data before s
throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to thrott
tls: # [object] TLS settings (client side)
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
  servername: # [string] Server name (SNI) - Server name for the SNI (Server Nam
  certificateName: # [string] Certificate name - The name of the predefined cert
  caPath: # [string] CA certificate path - Path on client in which to find CA ce
  privKeyPath: # [string] Private key path (mutual auth) - Path on client in whi
  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 - Minimum TLS version to use when c
  maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when c

  # ------------------------------------------------------

connectionTimeout: # [number] Connection Timeout - Amount of time (milliseconds)
writeTimeout: # [number] Write Timeout - Amount of time (milliseconds) to wait f
tokenTTLMinutes: # [number] Auth Token TTL minutes - The number of minutes befor
excludeFields: # [array of strings] Exclude Fields - Fields to exclude from the
onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or qu

# -------------- if onBackpressure is queue ---------------

pqMaxFileSize: # [string] Max file size - The maximum size to store in each queu
pqMaxSize: # [string] Max queue size - The maximum amount of disk space the queu
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 - Whether to block or drop even
pqControls: # [object]

# ------------------------------------------------------

pipeline: # [string] Pipeline - Pipeline to process data before sending out to t
systemFields: # [array of strings] System fields - Set of fields to automatical
environment: # [string] Environment - Optionally, enable this config only on a s
streamtags: # [array of strings] Tags - Add tags for filtering and grouping in S
```

```
cribl_http_output: # [object]
  type: # [string] Output Type
  loadBalanced: # [boolean] Load balancing - Use load-balanced destinations

  # -------------- if loadBalanced is false ---------------

  url: # [string] Cribl endpoint - URL of a Cribl Worker to send events to, e.g., l
  useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS loo

  # ------------------------------------------------------



  # -------------- if loadBalanced is true ---------------

  excludeSelf: # [boolean] Exclude current host IPs - Exclude all IPs of the curre
  urls: # [array] Cribl Worker Endpoints
    - url: # [string] Cribl Endpoint - URL of a Cribl Worker to send events to, e.g
      weight: # [number] Load Weight - The weight to use for load-balancing purpos
  dnsResolvePeriodSec: # [number] DNS resolution period (seconds) - Re-resolve any
  loadBalanceStatsPeriodSec: # [number] Load balance stats period (seconds) - How

  # ------------------------------------------------------

  tls: # [object] TLS settings (client side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are n
    servername: # [string] Server name (SNI) - Server name for the SNI (Server Name
    certificateName: # [string] Certificate name - The name of the predefined certi
    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 - Minimum TLS version to use when co
    maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when co

    # ------------------------------------------------------

  tokenTTLMinutes: # [number] Auth Token TTL minutes - The number of minutes before
  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] Request concurrency - Maximum number of ongoing requests
  maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the re
  maxPayloadEvents: # [number] Max events per request - Max number of events to in
  rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are no
  timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. S
  extraHttpHeaders: # [array] Extra HTTP headers - Extra HTTP headers.
    - name: # [string] Name - Field name
      value: # [string] Value - Field value
  failedRequestLoggingMode: # [string] Failed request logging mode - Determines wh
  safeHeaders: # [array of strings] Safe headers - List of headers that are safe t
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in S
  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or qu

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] Max file size - The maximum size to store in each queu
  pqMaxSize: # [string] Max queue size - The maximum amount of disk space the queu
  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 - Whether to block or drop even
  pqControls: # [object]

  # ----------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to t
  systemFields: # [array of strings] System fields - Set of fields to automatical
  environment: # [string] Environment - Optionally, enable this config only on a s
humio_hec_output: # [object]
  type: # [string] Output Type
  loadBalanced: # [boolean] Load balancing - Use load-balanced destinations

  # -------------- if loadBalanced is false ---------------

  url: # [string] Humio HEC Endpoint - URL to a Humio HEC endpoint to send events
  useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS lo

  # ----------------------------------------------------------


  # -------------- if loadBalanced is true ---------------

  excludeSelf: # [boolean] Exclude current host IPs - Exclude all IPs of the curre
```

```
urls: # [array] Humio HEC Endpoints
  - url: # [string] HEC Endpoint - URL to a Humio HEC endpoint to send events to
    weight: # [number] Load Weight - The weight to use for load-balancing purpos
dnsResolvePeriodSec: # [number] DNS resolution period (seconds) - Re-resolve any
loadBalanceStatsPeriodSec: # [number] Load balance stats period (seconds) - How

# -------------------------------------------------------

concurrency: # [number] Request concurrency - Maximum number of ongoing requests
maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the re
maxPayloadEvents: # [number] Max events per request - Max number of events to in
compress: # [boolean] Compress - Whether to compress the payload body before sen
rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are no
timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. S
extraHttpHeaders: # [array] Extra HTTP headers - Extra HTTP headers.
  - name: # [string] Name - Field name
    value: # [string] Value - Field value
failedRequestLoggingMode: # [string] Failed request logging mode - Determines wh
safeHeaders: # [array of strings] Safe headers - List of headers that are safe t
format: # [string] Request Format - Send data in JSON format to the api/v1/inges
authType: # [string] Authentication method - Enter a token directly, or provide

# -------------- if authType is manual ---------------

token: # [string] HEC Auth token - Humio HEC authentication token

# -------------------------------------------------------


# -------------- if authType is secret ---------------

textSecret: # [string] HEC Auth token (text secret) - Select or create a stored

# -------------------------------------------------------

onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or qu

# -------------- if onBackpressure is queue ---------------

pqMaxFileSize: # [string] Max file size - The maximum size to store in each queu
pqMaxSize: # [string] Max queue size - The maximum amount of disk space the queu
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 - Whether to block or drop even
pqControls: # [object]

# -------------------------------------------------------

pipeline: # [string] Pipeline - Pipeline to process data before sending out to t
systemFields: # [array of strings] System fields - Set of fields to automatical
environment: # [string] Environment - Optionally, enable this config only on a s
streamtags: # [array of strings] Tags - Add tags for filtering and grouping in S
```

15.4.16. Parsers.Yml

`parsers.yml` stores configuration data for the Knowledge > Parsers Library.

$CRIBL_HOME/default/cribl/parsers.yml

```
parser_id: # [object]
  lib: # [string] Library
  description: # [string] Description - Brief description of this parser. Optional.
  tags: # [string] Tags - One or more tags related to this parser. Optional.
  type: # [string] Type - Parser/Formatter type to use.
  fields: # [array of strings] List of Fields - Fields expected to be extracted, in
```

15.4.17. Perms.Yml

`perms.yml` stores Permissions configured for Cribl Edge users.

```
perms:
  - gid:    # [string]
    id:     # [string] Not used if type is 'groups'.
    type:   # [string] Resource type, one of: 'groups' | 'datasets' | 'dataset-provi
    policy: # [string] Policy name.
```

15.4.18. Policies.Yml

`policies.yml` contains RBAC Policy definitions. Default example:

$CRIBL_HOME/default/cribl/policies.yml

```
GroupFull:
  args:
    - groupName
  template:
    - PATCH /master/groups/${groupName}/deploy
    - GroupEdit ${groupName}
GroupEdit:
  args:
    - groupName
  template:
    - '* /m/${groupName}'
    - '* /m/${groupName}/*'
    - GroupRead ${groupName}
GroupCollect:
  args:
    - groupName
  template:
    - POST /m/${groupName}/lib/jobs
    - PATCH /m/${groupName}/lib/jobs/*
    - POST /m/${groupName}/jobs
    - PATCH /m/${groupName}/jobs/*
    - GroupRead ${groupName}
GroupRead:
  args:
    - groupName
  template:
    - GET /m/${groupName}
    - GET /m/${groupName}/*
    - POST /m/${groupName}/preview
    - POST /m/${groupName}/system/capture
    - POST /m/${groupName}/lib/expression
    - GET /master/groups/${groupName}
    - GET /master/workers
    - GET /master/workers/*
    - '* /w/*'
    - GET /master/groups
    - GET /system/info
    - GET /system/info/*
    - GET /system/logs
    - GET /system/logs/group/${groupName}/*
    - GET /system/settings
    - GET /system/settings/*
```

- GET /system/instance/distributed
- GET /system/instance/distributed/*
- GET /version
- GET /version/*
- GET /version/info
- GET /version/info/*
- GET /version/status
- GET /version/status/*
- GET /mappings
- GET /mappings/*
- GET /system/messages
- GET /system/message/*
- GET /ui/*
- POST /system/metrics/query
- GET /clui
- POST /system/capture

15.4.19. 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:

$CRIBL_HOME/default/cribl/regexes.yml

```
regex_id: # [object]
  lib: # [string] Library
  description: # [string] Description - Brief description of this regex. Optional.
  regex: # [string] Regex pattern - Regex pattern. Required.
  sampleData: # [string] Sample data - Sample data for this regex. Optional.
  tags: # [string] Tags - One or more tags related to this regex. Optional.
```

15.4.20. Roles.Yml

`roles.yml` contains RBAC Role definitions. Default example:

$CRIBL_HOME/default/cribl/roles.yml

```
admin:
  description: 'Members with admin role have permission to do anything and everythin
  policy:
    - '* *'
reader_all:
  description: 'Members with reader_all role get read-only access to all Worker Grou
  policy:
    - GroupRead *
collect_all:
  description: 'Members of this group can run existing collection jobs of all Worker
  policy:
    - GroupCollect *
editor_all:
  description: 'Members with editor_all role get read/write access to all Worker Gro
  policy:
    - GroupEdit *
owner_all:
  description: 'Members with owner_all role get read/write access as well as Deploy
  policy:
    - GroupFull *
user:
  description: 'The base user role allows users to see the system info along with th
  policy:
    - GET /system/info
    - GET /system/info/*
    - GET /system/users
    - GET /system/instance/distributed
    - GET /system/instance/distributed/*
    - GET /clui
    - PATCH /ui/*
```

15.4.21. Samples.Yml

`samples.yml` contains metadata about about stored sample data files (size, number of events, date created, name, etc.). Each sample is listed according to the following pattern:

$CRIBL_HOME/local/cribl/samples.yml

```
sample_id: # [object]
  sampleName: # [string] File Name - Filename to save the sample as. Required.
  pipelineId: # [string] Associate with Pipeline - Select a pipeline to associate wi
  description: # [string] Description - Brief description of this sample file. Optio
  ttl: # [number] Expiration (hours) - Time to live for the sample, the TTL is reset
  tags: # [string] Tags - One or more tags related to this sample file. Optional.
```

The corresponding sample files reside in `$CRIBL_HOME/data/samples`.

15.4.22. Schemas.Yml

`schemas.yml` stores configuration data for the Knowledge > Schema Library.

$CRIBL_HOME/default/cribl/schemas.yml

```
schema_id: # [object]
  description: # [string] Description - Brief description of this schema. Optional.
  schema: # [string] Schema - JSON schema matching standards of draft version 2019-0
```

15.4.23. Scripts.Yml

`scripts.yml` stores configuration data for scripts configured at **Settings** > **Global** > **Scripts**:

$CRIBL_HOME/local/cribl/scripts.yml

```
script_id: # [object]
  command: # [string] Command - Command to execute for this script
  description: # [string] Description - Brief description of this script. Optional.
  args: # [array of strings] Arguments - Arguments to pass when executing this scrip
  env: # [object] Env Variables - Extra environment variables to set when executing
```

15.4.24. Secrets.Yml

`secrets.yml` stores secrets for Cribl Edge.

The secrets are decrypted and encrypted using a `cribl.secret` file. `cribl.secret` is unique per Fleet and resides in the `$CRIBL_HOME/groups/<group-name>/local/cribl/auth` directory on Leader nodes, or in `$CRIBL_HOME/local/cribl/auth` for Workers or single-instance deployments.

Cribl automatically decrypts plaintext secrets from secrets.yml on startup and stores them in memory, keeping your data secure.

```
<secret-id>:
  secretType: # [string] One of: keypair | text | credentials
  secrets:
    # -------------- if secretType is keypair ---------------
    apiKey: "<encrypted-value>"
    secretKey: "<encrypted-value>"

    # -------------- if secretType is text ------------------
    value: "<encrypted-value>"

    # -------------- if secretType is credentials -----------
    username: "<encrypted-value>"
    password: "<encrypted-value>"
```

15.4.25. Service.Yml

`service.yml` maintains configuration for Cribl Edge service processes.

In the UI, you can configure them on the Leader at **Settings** > **Global** > **System** > **Service Processes** > **Services**.

Be careful about reducing the predefined limits. Extremely low values can prevent some Cribl Edge components from functioning.

$CRIBL_HOME/default/cribl/service.yml

```
connections: # [Object] - Heap memory limit for the connection listener processes.
  procs: # [number; default: 1, minimum: 1]
  memoryLimit: # [string; default: 2GB]
metrics: # [Object] - Heap memory limit for the metrics process
  procs: # [number; default: 1] – Single process; values other than 1 will be treate
  memoryLimit: # [string; default: 2GB]
notifications: # [Object] - Heap memory limit for the notifications process
  procs: # [number; default: 1] – Single process; values other than 1 will be treate
  memoryLimit: # [string; default: 2GB]
```

15.4.26. Vars.Yml

`vars.yml` stores configuration data for the Knowledge > Global Variables Library.

$CRIBL_HOME/default/cribl/vars.yml

```
variable_id: # [object]
  lib: # [string] Library
  description: # [string] Description - Brief description of this variable. Optional
  type: # [string] Type - Type of variable.
  value: # [string] Value - Value of variable
  tags: # [string] Tags - One or more tags related to this variable. Optional.
```

15.5. 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 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

15.5.1. C.Crypto – Data Encryption And Decryption

C.Crypto.decrypt()

```
Crypto.decrypt(value: string, escape: boolean = false, escapeSeq = '"'): string
```

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)
```

C.Crypto.encrypt()

```
Crypto.encrypt(value: Buffer | string, keyclass: number, keyId?: string, defaultVal?
```

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.

Examples

To encrypt the `customer` field using key with `keyId` 2, use:

```
C.Crypto.encrypt(customer, -1, 2)
```

To encrypt the same field with a key selected from the Key Class 3, without specifying a `keyId`:

```
C.Crypto.encrypt(customer, 3)
```

C.Crypto.createHmac()

```
Crypto.createHmac(value: string | Buffer, secret: string, algorithm: string = 'sha25
```

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.

Parameter	Type	Description
`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')
```

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'
```

15.5.2. C.Decode And C.Encode - Encoding And Decoding

C.Decode – Data Decoding Methods

C.Decode.base64()

```
Decode.base64(val: string, resultEnc: string = 'utf8'): string | Buffer | undefined
```

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')
```

C.Decode.gzip()

```
Decode.gzip(value: Buffer | string, encoding?: BufferEncoding) : string
```

Gunzips the supplied value.

Parameter	Type	Description
`value`	any	Value to gunzip.

Parameter	Type	Description
`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.

Examples

To decode a `hostname` field which had been encoded using `hex`, run:

```
C.Decode.gzip(hostname, 'hex')
```

C.Decode.hex()

```
Decode.hex(val: string): number
```

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.uri()

```
Decode.uri(val: string): string
```

Performs URI-decoding of the given string.

Parameter	Type	Description
`val`	string	Value to decode.

C.Decode.inflate()

```
Decode.inflate(value: Buffer | string, encoding: BufferEncoding = 'base64', isRaw =
```

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)
```

To additionally specify a different encoding to use, run:

```
C.Decode.inflate(_raw, 'hex')
```

C.Encode – Data Encoding Methods

`C.Encode.base64()`

```
Encode.base64(val: string | Buffer, trimTrailEq: boolean = false, encoding?: BufferEr
```

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)
```

To base64-encode the `hexValue` field, where the value of the field is a hex string:

```
C.Encode.base64('4672616e6b7575757575', undefined, 'hex')
```

C.Encode.gzip()

```
Encode.gzip(value: Buffer | string, encoding?: string) : string | Buffer
```

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')
```

C.Encode.hex()

```
Encode.hex(val: string | number): string
```

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.uri()

```
Encode.uri(val: string): string
```

Returns the URI-encoded representation of the given string.

Parameter	Type	Description
`val`	string	Value to encode.

C.Encode.deflate()

```
Encode.deflate(value: Buffer | string, encoding: BufferEncoding | 'none' = 'base64',
```

Deflates and optionally base64-encodes the supplied value.

Parameter	Type	Description
`value`	string | 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 value.

Examples

To deflate and hex-decode the value provided in an `inflated` field, run:

```
C.Encode.deflate(inflated, 'hex')
```

15.5.3. C.Lookup – Inline Lookup Methods

C.Lookup – Exact Lookup

```
Lookup: (file: string, primaryKey?: string, otherFields: string[]=[],ignoreCase: boo
```

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)
```

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_s
```

This example checks whether `someValue` is present or not and returns the answer as a boolean.

```
C.Lookup('file', 'primaryKeyCol', additionalCols).match('someValue')
```

This expression returns a string – the `fieldToReturn` column from the matched lookup row:

```
C.Lookup('file', 'primaryKeyCol', additionalCols).match('someValue','fieldToReturn')
```

This expression returns an array – the specified `fieldToReturn1` and `fieldToReturn2` columns from the matched lookup row:

```
C.Lookup('file', 'primaryKeyCol').match('someValue', ['fieldToReturn1', 'fieldToRetu
```

To return an array with all columns from the matched lookup row, you can use:

```
C.Lookup('file', 'primaryKeyCol').match('someValue', [])
```

`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 later support returning all columns from a matched row. Use the empty-array convention shown in the above examples: `.match('someValue',[])`.

C.LookupCIDR – CIDR Lookup

```
LookupCIDR: (file: string, primaryKey?: string, otherFields: string[]=[]) => InlineL
```

Returns an instance of a CIDR lookup to use inline.

C.LookupIgnoreCase – Case-insensitive Lookup

```
LookupIgnoreCase: (file: string, primaryKey?: string, otherFields: string[]=[]) => I
```

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

```
LookupRegex: (file: string, primaryKey?: string, otherFields: string[]=[]) => Inline
```

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}
```

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')
```

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')
```

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')
```

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')
```

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')
```

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`.

15.5.4. C.Mask – Data Masking Methods

C.Mask.CC()

```
Mask.CC(value: string, unmasked: number = -4, maskChar: string='X'): string
```

Checks whether a value could be a valid credit card number, and masks a subset of the value. By default, all digits except the last 4 will be replaced with `X`.

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

Let's assume 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.crc32()

```
Mask.crc32(value: string): string
```

Generates a CRC32 hash of given value.

Parameter	Type	Description
`value`	string	The string to hash.

C.Mask.IMEI()

```
Mask.IMEI(value: string, unmasked: number = -4, maskChar: string='X'): string
```

Checks whether a value could be a valid IMEI number, and masks a subset of the value. By default, all digits except the last 4 will be replaced with `X`.

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

Let's assume 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.CC(g1, 4, "M")}`	`cardNumber: 5356MMMMMMMMMM`

C.Mask.isCC()

```
Mask.isCC(value: string): boolean
```

Checks whether the given value could be a valid credit card number, by computing the string's Luhn's checksum modulo 10 == `0`.

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 : ""
```

C.Mask.isIMEI()

```
Mask.isIMEI(value: string): boolean
```

Checks whether the given value could be a valid IMEI number, by computing the string's Luhn's checksum modulo 10 == `0`.

Parameter	Type	Description
`value`	string	String to check for being a valid IMEI number.

Examples

The following example expression uses `isCC()` to check whether the `imei` field actually contains ab 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 : ""
```

C.Mask.luhn()

```
Mask.luhn(value: string, unmasked: number = -4, maskChar: string='X'): string
```

Checks that value Luhn's 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.

Parameter	Type	Description
`value`	string	The string to mask, **if and only if** the value's Luhn's 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.

C.Mask.luhnChecksum()

```
Mask.luhnChecksum(value: string, mod: number=10): number
```

Generates the Luhn checksum (used to validate certain credit card numbers, IMEIs, etc.). By default, the mod 10 of the checksum is returned. Pass mod = `0` to get the actual checksum.

Parameter	Type	Description
`value`	string	String whose digits you want to perform the Luhn checksum on.
`mod`	number	Return checksum modulo this number. If `0`, skip modulo. Default is `10`.

C.Mask.md5()

```
Mask.md5(value: string, len?: string | number, encoding?: BufferEncoding): string
```

Generates an MD5 hash of a given value.

`C.Mask.md5()` doesn't work when Cribl Stream is running in FIPS mode.

Parameter	Type	Description
`value`	string	The string to hash.
`len`	string | 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 it's 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)
```

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')
```

C.Mask.random

```
Mask.random(len?: string | number): string
```

Generates a random alphanumeric string.

Parameter	Type	Description
`len`	string | 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)
```

Input	Output
`McGoatFace`	`xY2KLHkyTg`

C.Mask.repeat()

```
Mask.repeat(len?: number | string, char: string = 'X'): string
```

Generates a repeating char/string pattern, e.g., `XXXX`.

Parameter	Type	Description
`len`	string | 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")
```

C.Mask.sha1(), C.Mask.sha256(), C.Mask.sha512()

```
Mask.sha1(value: string, len?: string | number, encoding?: BufferEncoding): string
Mask.sha256(value: string, len?: string | number, encoding?: BufferEncoding): string
Mask.sha512(value: string, len?: string | number, encoding?: BufferEncoding): string
```

Generates an SHA1, SHA256, or SHA512 hash of given value.

Parameter	Type	Description
`value`	string	The string to hash.
`len`	string | 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)
```

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')
```

C.Mask.REDACTED

```
Mask.REDACTED: string
```

The literal `'REDACTED'`.

15.5.5. C.Net – Network Methods

C.Net.cidrMatch()

```
Net.cidrMatch(cidrIpRange: string, ipAddress: string): boolean
```

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()

```
Net.communityIDv1(sourceIP: string, destIP: string, sourcePort: number | undefined, 
```

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 Edge NetFlow 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='
```

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='
```

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='
```

C.Net.isIp()

```
Net.isIp(value: string): boolean
```

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()

```
Net.isIpV4(value: string): boolean
```

Determines whether the value supplied is an IPv4 address.

Parameter	Type	Description
`value`	string	The IP address to test.

C.Net.isIpV6()

```
Net.isIpV6(value: string): boolean
```

Determines whether the value supplied is an IPv6 address.

Parameter	Type	Description
`value`	string	The IP address to test.

C.Net.isIPV4Cidr()

```
Net.isIPV4Cidr(value: string): boolean
```

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()

```
Net.isIPV6Cidr(value: string): boolean
```

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.

Parameter	Type	Description

C.Net.isIpV4AllInterfaces()

```
Net.isIpV4AllInterfaces(value): boolean
```

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()

```
Net.isIpV6AllInterfaces(value): boolean
```

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()

```
Net.ipv6Normalize(address: string): string
```

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()

```
Net.isPrivate(address: string): boolean
```

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()

```
Net.parseAddressOrRangeString(source: string, mask?: string | number)
```

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

```
Net.IPv4_CIDR_REGEX: RegExp
```

The regular expressions for matching IP CIDR.

15.5.6. C.Text – Text Methods

C.Text.entropy()

```
Text.entropy(bytes: Buffer | string): number
```

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()

```
Text.hashCode(val: string | Buffer | number): number
```

Computes hashcode (djb2) of the given value.

Returns hashcode value.

Parameter	Type	Description
`val`	string | Buffer | number	value to be hashed.

C.Text.isASCII()

```
Text.isASCII(bytes: Buffer | string): boolean
```

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()

```
Text.isUTF8(bytes: Buffer | string): boolean
```

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()

```
Text.parseWinEvent(xml: string, nonValues: string[] = Text._WIN_EVENT_NON_VALUES): a
```

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.
`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()

```
Text.parseXml(xml: string, keepAttr: boolean = true, keepMetadata: boolean = false, 
```

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()

```
Text.relativeEntropy(bytes: Buffer | string, modelName: string = 'top_domains'): numl
```

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.

15.5.7. C.Time – Time Methods

C.Time.adjustTZ()

```
Time.adjustTZ(epochTime: number, tzTo: string, tzFrom: string = 'utc'): number | und
```

Adjusts a timestamp from one timezone to another.

Returns the adjusted timestamp, in UNIX epoch time (ms).

Parameter	Type	Description
`epochTime`	number	Timestamp to adjust, in UNIX epoch 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")
```

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 epoch time in seconds, you can divide it by 1000:

```
C.Time.strftime(C.Time.adjustTZ(_time, "Australia/Melbourne") / 1000, "%d %B %Y %H %I
```

C.Time.clamp()

```
Time.clamp(date: T, earliest: T, latest: T, defaultDate?: T ): T  | undefined
```

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 epoch time (ms) or JavaScript Date format.
`earliest`	Date | number	Earliest allowable timestamp, in UNIX epoch time (ms) or JavaScript Date format.
`latest`	Date | number	Latest allowable timestamp, in UNIX epoch time (ms) or JavaScript Date format.
`defaultDate` (optional)	Date | number	Default date, in UNIX epoch 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")
```

Input	Output
`"2016-02-11"`	`"2020-01-01"` (set to the earliest value)
`"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")
```

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()

```
Time.strftime(date: number | Date | string, format: string, utc: boolean = true): st
```

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 epoch 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 UNIX timestamp 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")
```

Some common formats (assuming UNIX timestamp 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"`

Format	Output
`"%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()

```
Time.strptime(str: string, format: string, utc: boolean = true, strict: boolean = fa
```

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.
`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")
```

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)
```

For detailed reference of the date and time tokens, see d3js reference or Auto Timestamp.

C.Time.timestampFinder()

```
Time.timestampFinder(utc?: boolean).find(str: string): IAutoTimeParser
```

Extracts time from the specified field, using the same algorithm as the Auto Timestamp Function and the Event Breaker Function.

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)
```

You can then format the string output with `strftime()`:

```
C.Time.strftime(C.Time.timestampFinder(true).find(_raw), "%d.%m.%Y, %H.%M")
```

C.Time.timePartition()

```
Time.timePartition(date: number | Date | string, level: 'd' | 'h'): string | undefin
```

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)
```

This will return the prefix in the form of, for example: `2024/04/23`.

C.Time.s3TimePartition()

```
Time.s3TimePartition(date: number | Date | string, level: 'd' | 'h'): string | undef
```

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)
```

This will return the prefix in the form of, for example: `hjhg-2024/aj-04/ai-23`.

15.5.8. Miscellaneous Expression Methods

C.Schema – Schema Methods

C.Schema

```
Schema: (id: string) => SchemaValidator
```

C.Schema.validate()

```
SchemaValidator.validate(data: any): boolean
```

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)
```

See Schema Library for more details.

C.Secret – Secrets-Management Methods

C.Secret()

```
Secret: (id: string, type?: string): ISecret
Secret(id: string, type: 'keypair') => IPairSecret
Secret(id: string, type: 'text') => ITextSecret
Secret(id: string, type: 'credentials') => ICredentialsSecret
```

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')
```

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
```

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`

```
env: {[key: string]: string;}
```

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
```

To return the hostname of the machine where Cribl Edge is running, use:

```
C.env.HOSTNAME
```

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.

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()

```
Misc.zip(keys: string[], values: any[], dest?: any): any
```

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)
```

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
```

Returns a version 4 (random) UUID in accordance with RFC-4122.

Examples

To create a UUIDv4, use:

```
C.Misc.uuidv4()
```

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
```

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')
```

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
```

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())
```

Result: returns `true`

```
C.Misc.validateUUID('clearly not a UUID')
```

Result: returns `false`

C.Misc.getUUIDVersion()

```
C.Misc.getUUIDVersion(uuid: string): number
```

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())
```

Result: `4`

```
C.Misc.getUUIDVersion(C.Misc.uuidv5('example', 'X500'))
```

Result: `5`

15.6. Cribl Expression Syntax

As data travels through a Cribl Edge Pipeline, it is operated on by a series of Functions. Functions are fundamentally JavaScript code.

Functions that ship with Cribl Edge are configurable via a set of inputs. Some of these configuration options are literals, such as field names, and others can be JavaScript expressions.

Expressions are **valid units** of code that resolve to a value. Every syntactically valid expression resolves to some value, but conceptually, there are two types of expressions: those that **assign** value to a variable (a.k.a., with side effects), and those that **evaluate** to a value.

Assigning a value	Evaluating to a value

`x = 42`
`newFoo = foo.slice(30)`
	`(Math.random() * 42)`
	`3 + 4`
	`'foobar'`
	`'42'`

Filters and Value Expressions

Let's consider evaluation expressions before assignment expressions.

Filters

Filters are used in Routes to select a stream of the data flow, and in Functions to scope or narrow down the applicability of a Function. Filters are expressions that **must** evaluate to either `true` (or truthy) or `false` (or falsy). Keep this in mind when creating Routes or Functions. For example:

- `sourcetype=='access_combined' && host.startsWith('web')`
- `source.endsWith('.log') || sourcetype=='aws:cloudwatchlogs:vpcflow'`

This table shows examples of truthy and falsy values.

Truthy	Falsy

`true`
`42`
`-42`
`3.14`
`"foo"`
	`false`
	`null`
	`undefined`
	`0`
	`NaN`

Truthy	Falsy
`Infinity` `-Infinity`	`''` `""`

Value Expressions

Value expressions are typically used in Functions to assign a value – for example, to a new field. For example:

- `Math.floor(_time/3600)`
- `source.replace(/.{3}/, 'XXX')`

Best Practices for Creating 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 – e.g., `@timestamp` or `user-agent` or `kubernetes.namespace_name` – you can access them using `__e['<field-name-here>']`. (Note the single quotes.) More details here.

In any other place where the field is referenced – e.g., in the Eval function's field names – you should use a single-quoted literal, of the form: `'<field-name-here>'`.

Wildcard Lists

Wildcard Lists are used throughout the product, especially in various Functions, such as Eval, Mask, Publish Metrics, Parser, etc.

Wildcard Lists, as their name implies, 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 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:

Wildcard List	Value	Meaning
List 1	`!foobar, foo*`	All terms that start with **foo**, except **foobar**.
List 2	`!foo*, *`	All terms, except for those that start with **foo**.
List 3	`*, !foo`	All terms (wildcard matches first, negation isn't evaluated).

You cannot use wildcards to target Cribl Edge internal fields that start with `__` (double underscore). You must specify these fields individually. For example, `__foobartab` cannot be removed by specifying `__foo*`.

15.7. 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 AppScope 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. Defaults to `/var/run/secrets/kubernetes.io/serviceaccount`.

Name	Purpose
`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.

Distributed Deployment Variables

You can use the following environment variables to configure your distributed Cribl Edge instance.

Name	Purpose
`CRIBL_DIST_MASTER_URL`	URL of the Leader Node. Example: `CRIBL_DIST_MASTER_URL=tls://<authToken>@leader:4200` See Usage Notes below.
`CRIBL_DIST_MODE`	`worker` or `master`. Defaults to `worker`, if (and only if) `CRIBL_DIST_MASTER_URL` is present.
`CRIBL_DIST_WORKER_PROXY`	Communicate to the Leader Node via a SOCKS proxy. See Usage Notes below.

Usage Notes

This section explains how to use certain complex environment variables.

`CRIBL_DIST_MASTER_URL`

Use this format:

```
<tls|tcp>://<authToken>@host:port?group=defaultGroup&tag=tag1&tag=tag2&tls.
<tls_settings>
```

The auth token can include only alphanumeric characters or the underscore (`_`). Do not include `<`, `>`, `"`, `` ` ``, `\r`, `\n`, `\t`, `{`, `}`, `|`, `\`, `^`, or `'`.

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`.

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.

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 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:
    ...
```

For details about each file's syntax, see Config Files and its child topics.

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` m
`CRIBL_DIST_MASTER_FAILOVER_VOLUME=/tmp/shared`	Sets the location of the NFS directory to support failover.
`CRIBL_DIST_MASTER_FAILOVER_MISSED_HB_LIMIT`	Determines how many Lease refresh periods ela the standby Nodes attempt to promote themsel primary. Cribl recommends setting this to `3`.
`CRIBL_DIST_MASTER_FAILOVER_PERIOD`	Determines how often the primary Leader refre 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 volun to access `$CRIBL_INSTANCE_HOME/local/_system/in` (`C:\ProgramData\Cribl\local\_system.yr` 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.
`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

Name	Purpose
	`$CRIBL_VOLUME_DIR/state/spool`.

CRIBL_HOME

`CRIBL_HOME` is a special internal environment variable that indicates the location of the Cribl binary (`bin` directory). You do not set it manually, but it is referred to in multiple places and commands.

15.8. Linux System Metrics Details

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	Counter	`release`, `sysname`, `version`	Basic

Name	Description	Type	Dimensions	Mode
	by the uname system call.			
`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

Name	Description	Type	Dimensions	Mode
`node_vmstat_pgpgout`	`/proc/vmstat` information field `pgpgout`.	Gauge	N/A	Basic
`node_vmstat_pswpin`	`/proc/vmstat` information field `pswpin`.	Gauge	N/A	Basic
`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_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

Name	Description	Type	Dimensions	Mo
node_network_receive_errs_all_total	Network device statistic receive_errs.	Counter	N/A	Bas
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

Name	Description	Type	Dimensions	Mo
node_network_transmit_packets_total	Network device statistic transmit_packets per interface.	Counter	device	Cus Per Int
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	Counter	N/A	Cus Det

Name	Description	Type	Dimensions	Mo
	connections.			Met
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

Name	Description	Type	Dimensions
node_disk_writes_completed_all_total	Network device statistic receive_packets.	Counter	N/A
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

Name	Description	Type	Dimensions
node_disk_read_bytes_total	Total number of bytes read successfully per disk.	Counter	device, fstype, mountpoint
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

Name	Description	Type	Dimensions
node_disk_writes_merged_all_total	Total number of writes merged.	Counter	N/A
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

Name	Description	Type	Dimensions
`node_filesystem_avail_bytes`	Filesystem space available to non-root users in bytes per disk.	Gauge	`device`, `fstype`, `mountpoint`
`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 since the epoch) when the container was started.	Counter
`container_finish_time_seconds`	UNIX time (seconds since the epoch) 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

Name	Description	Type
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
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

Name	Description	Type
`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	Counter	`process_cmdline`, `process_set`, `process_uid`, `process_gid`, `process_service`

Name	Description	Type	Dimensions
	block-backed filesystems.		
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
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-	Gauge	process_cmdline, process_set, process_uid, process_gid, process_service

Name	Description	Type	Dimensions
	loaded in or are swapped out.		
process_virtual_memory_bytes	The process's virtual memory size.	Gauge	process_cmdline, process_set, process_uid, process_gid, process_service
process_swapped_memory_bytes	The process's swapped memory size.	Gauge	process_cmdline, process_set, process_uid, process_gid, process_service
process_memory_bytes	The total amount of memory used by the process, in bytes.	Gauge	process_cmdline, process_set, process_uid, process_gid, process_service
process_memory_usage	The total amount of memory used by the process, as a percentage of total memory.	Gauge	process_cmdline, process_set, process_uid, process_gid, process_service
process_start_time	The time that the process started, derived by adding the start time to the boot time, making it relative to epoch.	Gauge	process_cmdline, process_set, process_uid, process_gid, process_service

15.9. Windows System Metrics Details

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

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	Mod
`windows_cs_logical_processors`	Number of installed	Gauge	N/A	Basic

Name	Description	Type	Dimensions	Mod
	logical processors.			
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
windows_cpu_info	Labeled CPU information.	Gauge	architecture, device_id, description, family,	Custo Deta

Name	Description	Type	Dimensions	Mod
			`l2_cache_size`, `l3_cache_size`, `name`	
`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
`windows_os_virtual_memory_bytes`	Bytes of virtual memory.	Gauge	N/A	Custo Deta
`windows_system_exception_dispatches_total`	Total exceptions	Counter	N/A	Custo Deta

Name	Description	Type	Dimensions	Mod
	dispatched by the system.			
`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	Mod
`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 time metri
`windows_cpu_percent`	CPU percent active usage	Gauge	`core`, `mode` is set to `user`, `idle`, `privileged`, `interrupt`, `dpc`	Basic Custo Per and time metri

Name	Description	Type	Dimensions	Mod
windows_cpu_parking_status	Parking Status represents whether a processor is parked or not.	Counter	core	Basic Custo Per and time metri
windows_cpu_core_frequency_mhz	Core frequency in megahertz.	Gauge	core	Basic Custo Per and 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 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 received and serviced deferred procedure calls (DPCs).	Counter	core	Custo Per or Deta metri

Name	Description	Type	Dimensions	Mod
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 work the processor could complete if it were running at	Gauge.	core	Custo Per and Deta metri

Name	Description	Type	Dimensions	Mod
	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 received and serviced clock tick interrupts, computed average on a	Gauge	mode	Custo Per and Deta metri

Name	Description	Type	Dimensions	Mod
	per second interval.			
`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`	Custo `Per` and `Deta` metri
`windows_cpu_percent_all_total`	Core frequency in megahertz.	Gauge	`mode` is set to `dpc`, `idle`, `interrupt`, `privilege`, `user`,`active`	Custo 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	Dimensi
`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
`windows_memory_cache_bytes`	Number of bytes currently being used by the filesystem cache	Gauge	N/A

Name	Description	Type	Dimensi
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 portion of the paged pool that is currently resident and active in physical	Gauge	N/A

Name	Description	Type	Dimensi
	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
windows_memory_modified_page_list_bytes	Physical memory, in bytes, assigned to the modified page list. This	Gauge	N/A

Name	Description	Type	Dimensi
	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.		
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 single operation).	Counter	N/A
windows_memory_swap_page_operations_total	Total number of swap page read	Counter	N/A

Name	Description	Type	Dimensi
	and writes (PagesPersec).		
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 core standby cache page lists. This memory contains cached data and code	Gauge	N/A

Name	Description	Type	Dimensi
	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 memory, memory on lower priority standby cache page lists will be	Gauge	N/A

Name	Description	Type	Dimensi
	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
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	Gauge	N/A

Name	Description	Type	Dimensi
	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.		
`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, 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.	Gauge	N/A
`windows_memory_system_driver_resident_bytes`	The size, in bytes, of the	Gauge	N/A

Name	Description	Type	Dimensi
	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.		
`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 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.	Gauge	N/A

Name	Description	Type	Dimensi
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 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	Counter	N/A

Name	Description	Type	Dimensi
	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 memory used.	Gauge	N/A
windows_memory_available_percent	Percent 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 computed per sec.	Gauge	N/A
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

Name	Description	Type	Dimensi
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 in physical memory computed per sec.	Gauge	N/A

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	Dimer
`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 detected to prevent delivery.	Counter	`nic`
`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	Counter	`nic`

Name	Description	Type	Dimer
	discarded because of an unknown or unsupported protocol.		
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_unicast_total	Total subnet-unicast packets that are delivered to a higher-layer protocol.	Counter	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

Name	Description	Type	Dimer
windows_net_bytes_received_total	Total bytes received by interface.	Counter	nic
wwindows_net_bytes_received_total_per_sec	Total bytes received by interface computed per sec.	Gauge	nic
windows_net_bytes_sent_total	Total bytes transmitted by interface.	Counter	nic
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

Name	Description	Type	Dimer
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_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 by interface.	Counter	nic
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

Name	Description	Type	Dimer
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 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.	Counter	af
windows_tcp_connections_active_all_total	Number of times TCP connections have made a	Counter	af

Name	Description	Type	Dimer
	direct transition from the CLOSED state to the SYN-SENT state.		
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
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	Counter	af

Name	Description	Type	Dimer
	using the TCP protocol.		
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 received using the TCP protocol computed per sec.	Gauge	af
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

Name	Description	Type	Dimer
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
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	Gauge	af

Name	Description	Type	Dimer
	the destination port.		
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 port computed per sec	Gauge	af
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

Name	Description	Type	Dimer
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	Counter	volume

Name	Description	Type	Dimensio
	on the disk.		
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 operation from the disk.	Counter	volume
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	Counter	volume

Name	Description	Type	Dimensio
	read/write requests).		
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_time	Percent time the disk was idle.	Gauge	volume
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	Gauge	volume

Name	Description	Type	Dimensio
	physical disk.		
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 transferred to or from the disk during read operations per sec.	Gauge	volume
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	Gauge	volume

Name	Description	Type	Dimensio
	logical disk per sec.		
windows_logical_disk_write_bytes_per_se	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_set, process_ppid, process_pid, process_cmdline, process_exe_path, process_service_name
process_cpu_time_total	Elapsed time that the process's threads have spent executing instructions in either privileged mode or	counter	process_set, process_ppid, process_pid, process_cmdline, process_exe_path, process_service_name

Name	Description	Type	Dimensions
	user mode. Included in this count is code executed to handle some hardware interrupts and trap conditions.		
`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_set`, `process_ppid`, `process_pid`, `process_cmdline`, `process_exe_path`, `process_service_name`
`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_set`, `process_ppid`, `process_pid`, `process_cmdline`, `process_exe_path`, `process_service_name`
`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_set`, `process_ppid`, `process_pid`, `process_cmdline`, `process_exe_path`, `process_service_name`

Name	Description	Type	Dimensions
process_page_faults_total	Total number of page faults by threads executing in this process. A page fault 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.	counter	process_set, process_ppid, process_pid, process_cmdline, process_exe_path, process_service_name
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_set, process_ppid, process_pid, process_cmdline, process_exe_path, process_service_name
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	gauge	process_set, process_ppid, process_pid, process_cmdline, process_exe_path, process_service_name

Name	Description	Type	Dimensions
	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.		
`process_priority_base`	Current base priority of this process. Threads within a process can raise and lower their own base priority, relative to the process's base priority.	gauge	`process_set`, `process_ppid`, `process_pid`, `process_cmdline`, `process_exe_path`, `process_service_name`
`process_private_bytes`	Current number of bytes this process has allocated that can't be shared with other processes.	gauge	`process_set`, `process_ppid`, `process_pid`, `process_cmdline`, `process_exe_path`, `process_service_name`
`process_threads`	Number of threads currently active in this process. Every running process has at least one thread.	gauge	`process_set`, `process_ppid`, `process_pid`, `process_cmdline`, `process_exe_path`, `process_service_name`
`process_virtual_bytes`	Current size, in bytes, of the virtual address space that the process is using. Use of virtual space doesn't imply use of either disk or main	gauge	`process_set`, `process_ppid`, `process_pid`, `process_cmdline`, `process_exe_path`, `process_service_name`

Name	Description	Type	Dimensions
	memory pages. Virtual space is finite and when the process uses too much, it can limit its ability to load libraries.		
process_working_set_private_byte	Size of the working set, in bytes, that is used for this process only and not shared or shareable by other processes.	gauge	process_set, process_ppid, process_pid, process_cmdline, process_exe_path, process_service_name
process_working_set_peak_bytes	Maximum size, in bytes, of the working set of this process at any point in time. The working set is the set of memory pages touched recently by the threads in the process. If free memory is above a threshold, pages are left in the working set of a process even if they are not in use. When free memory falls below a threshold, pages are trimmed from working sets. If pages are needed, they will be soft-faulted back into the working set before they leave main memory.	gauge	process_set, process_ppid, process_pid, process_cmdline, process_exe_path, process_service_name
process_working_set_bytes	Maximum number of bytes in the working set of this process at any point in time. If free memory is above a threshold, pages are left in the working set of a process	gauge	process_set, process_ppid, process_pid, process_cmdline, process_exe_path, process_service_name

Name	Description	Type	Dimensions
	even if they aren't in use. When free memory falls below a threshold, pages are trimmed from working sets. If pages are needed, they will be soft-faulted back into the working set before they leave main memory.		

16. Troubleshooting

16.1. Known Issues

Learn about issues affecting Cribl Stream and/or Cribl Edge. Find out if they're already fixed, or discover temporary workarounds.

In the tables below, "Reported in" refers to the product version in which the given issue was first recorded. In some cases, you might encounter that issue in an older version as well.

See also Known Issues resolved in older product versions:

- Resolved in 4.9
- Resolved in 4.8
- Resolved in 4.7
- Resolved in 4.6
- Resolved in 4.5
- Resolved in 4.4
- Resolved in 4.3
- Resolved in 4.2
- Resolved in 4.1
- Resolved in 4.0

Unresolved

Scheduled jobs in Single-instance mode do not resume when orphaned

Reported in	Resolved in	Ticket
4.9.0	TBD	CRIBL-29124

Problem: In Single-instance mode of Cribl Stream, orphaned scheduled jobs do not resume, resulting in stalled execution. This issue does not affect deployments in Distributed mode.

Workaround: None identified.

Erroneous FIPS mode detection in Cribl Edge on Windows

Reported in	Resolved in	Ticket
4.8.2	TBD	CRIBL-28927

Problem: Cribl Edge on Windows machines might encounter issues booting up when the `OPENSSL_MODULES` and `OPENSSL_CONF` environment variables are set.

Workaround: Use the Windows Registry to set the following environment variables to empty strings in the environment where the Cribl Edge service runs. For example:

```
CRIBL_VOLUME_DIR=C:\ProgramData\Cribl\
OPENSSL_CONF=
OPENSSL_MODULES=
```

Note: Upgrading Cribl Edge or running MSI repair will revert the workaround implemented in the registry. You will need to reapply the changes after the upgrade or repair.

Cribl server fails to start with AWS KMS and AssumeRole enabled

Reported in	Resolved in	Ticket
4.3.0	TBD	CRIBL-28922

Problem: When using AWS KMS with AssumeRole enabled, the Cribl server fails to start due to an error when accessing the `getCredentialsAsync` function. The error is logged as `boot failed, checking for rollback` in the `cribl.log` file.

Workaround: None identified.

Encrypted secrets are broken when using encoding other than `latin1`

Reported in	Resolved in	Ticket
2.1.0	TBD	CRIBL-28901

Problem: Secrets that contain characters in encodings other than `latin1` will return an incorrect value when decoded.

Workaround: Make sure your secrets use `latin1` encoding.

When you teleport into a Windows Edge Node and navigate to Health, the CPU Load Average panel displays no results.

Reported in	Resolved in	Ticket
4.6.1	TBD	CRIBL-25879

Problem: In Windows deployments of Cribl Edge, the **CPU Load Average** panel is blank when you teleport into a Node and view the **Health** page.

Workaround: None identified.

Dashboard UI charts missing metrics after application update

Reported in	Resolved in	Ticket
4.7.1	TBD	CRIBL-25601

Problem: After upgrading the application, dashboard charts may fail to display metrics recorded prior to the upgrade. This concerns the following charts:

- **Events In and Out**
- **Bytes In and Out**
- **Free Memory**
- **CPU Load Average**

This issue is intermittent and only affects visualization in the dashboard charts. Underlying data is flowing correctly.

Workaround: None identified.

Stopping a container does not initiate clean shutdown

Reported in	Resolved in	Ticket
4.6.1	TBD	CRIBL-25241

Problem: Stopping a container does not issue a graceful shutdown signal SIGTERM to the Cribl process. Containers apply their own timeouts for shutdown, and these override any application-level shutdown timeouts.

Workaround: None identified.

Empty directories that contained Parquet files are not deleted

Reported in	Resolved in	Ticket
4.3.1	TBD	CRIBL-23495

Problem: When certain Sources and Collectors ingest or collect Parquet files, they create directories for storing those files. After they send and delete the files, they should also delete the directories, but fail to do that, potentially consuming large numbers of inodes on the storage volume. Sources affected: Azure Blob Storage, Amazon S3. Collectors affected: Amazon S3, Filesystem/NFS, Google Cloud Storage.

Workaround: Configure a cron job to delete the empty directories.

After restarting the File Monitor Source, Event Breakers aren't processing CSV files correctly when the File Monitor Source is configured to collect CSV logs

Reported in	Resolved in	Ticket
4.4.4	TBD	CRIBL-22360

Problem: When the File Monitor Source is configured to collect CSV file logs and use CSV Event Breakers, subsequent event breaking is not working when the Source resets. The Event Breaker expects the first line of column headers, but they resume reading from a later point in the file where those headers aren't present.

Workaround: Use the default line breaker and a pre-processing Pipeline to drop lines that start with `#`. Optionally, use an `Eval` function to split `_raw` values on the column-separator character into separate fields and rename them as needed.

With Leader HA/Failover, creating diag bundles via CLI produces incorrect results

Reported in	Resolved in	Ticket
4.4.0	TBD	CRIBL-21258, CRIBL-21557

Problem: For deployments configured for Leader high availability/failover, creating diagnostic bundles via the UI works normally but creating them via the CLI does not. In Cribl Edge 4.4.0 and newer, the operation fails with a `not a git repository` error. In 4.4.0 and older, the diag bundle created contains obsolete data. In both cases the underlying cause is that the process that creates the diag bundle cannot access the `CRIBL_CONF_DIR` environment variable.

Workaround: Use the UI to create the diag bundle. Or, if you use the CLI, preface the `diag` command with the needed environment variable, like this: `CRIBL_CONF_DIR=/<path_to_failover_volume> cribl diag create`. If you have an `instance.yml` config file, you can find the path to the failover volume in the `distributed` section.

Fix: Error message improvements were made as part of CRIBL-21258. Fix for CRIBL-21557 is TBD.

Migrated Local Users appear in the Members UI with "No Access"

Reported in	Resolved in	Ticket
4.2.0	TBD	CRIBL-18973

Problem: Local Users are incorrectly shown in the **Settings** > **Members** UI with `No Access`. However, their Roles still function as originally configured, and still display correctly at **Settings** > **Global Settings** > **Access Management** > **Local Users**.

Workaround: Rely on the **Local Users** UI.

Fleet Upgrade Errors

Reported in	Resolved in	Ticket
4.1.3	TBD	CRIBL-18374

Problem: A Fleet's upgrade from a Leader can result in errors when the Edge Nodes use different `host/port/tls` settings than Worker Nodes.

Workaround: The Edge Nodes and Workers must connect to the leader using the `host/port/tls` connection details. If this is not possible, upgrade Edge Nodes separately using the boostrap script.

Datadog Agent Source fails to receive metrics

Reported in	Resolved in	Ticket
3.5.2	TBD	CRIBL-18281

Problem: When a Datadog Agent using Datadog API v2 sends metrics, Datadog Agent Source fails to receive them, because the Source uses Datadog API v1. Datadog Agent logs will show `API Key invalid, dropping transaction` errors. Debug logging will likely produce `Dropping request because of unallowed path message` errors with `statusCode: 403`, which Datadog Agent interprets as "invalid API key."

Workaround: In the `datadog.yaml` file, ensure that `use_v2_api.series` is set to `false`. Cribl Edge's Datadog Agent Source will then receive metrics normally, since both the Source and the Datadog Agent will be using Datadog API v1, which Datadog still supports.

Cannot clear Messages drawer while in GitOps Push mode

Reported in	Resolved in	Ticket
3.5.0	TBD	CRIBL-15239

Problem: When in GitOps `Push` mode, you cannot clear messages from the **Messages** drawer. Cribl Edge returns a `Forbidden` error.

Workaround: None identified.

Amazon Kinesis Data Streams Source resumes reading at earliest event when Leader Node is down

Reported in	Resolved in	Ticket
4.0.0	TBD	CRIBL-15059

Problem: After the Worker on which it is running restarts, the Kinesis Streams Source should resume reading a shard from the point where it left off. That point is a sequence number obtained from the Leader Node – but if the Leader Node is down, sequence numbers (state) will not be available. In this situation, the Worker should wait for the Leader Node to come back up, obtain the state, and then resume reading from the correct point in the shard.

However, due to this issue, the Worker does not wait for the Leader to come back up if the Leader stays down for over one minute. Instead, the Worker starts where **Optional Settings > Shard iterator start** says it should.

- If **Shard iterator start** is set to its default of `Earliest record`, the Worker will resume reading from the beginning of the shard, potentially collecting events it already collected earlier, producing duplicate events.
- If **Shard iterator start** is set to `Latest record`, the Worker will resume reading from the end of the shard, potentially missing events if the last event it collected before the restart is not the last event in the shard.

Workaround: None identified.

Recently added Worker Nodes fail to appear on the Monitoring page

Reported in	Resolved in	Ticket
3.5.0	TBD	CRIBL-14627

Problem: When Worker Nodes are added, there may be a delay before the Worker count is updated on the Monitoring page.

Workaround: Refresh your web browser.

Default commit message missing for non-admin users

Reported in	Resolved in	Ticket
4.0.0	TBD	CRIBL-14239

Problem: For users who have Roles as high as `owner_all`, but not `admin`, the Commit modal fails to display any **Default commit message** saved in **Git Settings**.

Workaround: Enter (or paste) a message per commit.

Diag bundles might fail to download from teleported Edge Nodes

Reported in	Resolved in	Ticket
4.1.0	TBD	CRIBL-13999

Problem: When you're remotely accessing a Edge Node's UI, diag bundles might fail to download from **Global Settings** > **Diagnostics**.

Workaround: Refresh the page and **Export** the bundle again. Alternatively, log directly into the Edge Node's UI before creating the diag.

Kubernetes Logs Source doesn't collect logs from exited or short-lived containers

Reported in	Resolved in	Ticket
4.0.0	TBD	CRIBL-13404

Problem: The Kubernetes Logs Source collects logs only from running containers, which can lead to missed logs from short-lived containers or those that exit before they start running.

Workaround: None identified.

Pipelines with Clone and other Functions can show inconsistent total processing times

Reported in	Resolved in	Ticket
4.0.0	TBD	CRIBL-13125

Problem: For Pipelines containing both a Clone Function and an async Functions (like Redis), the Pipeline Diagnostics modal's **Pipeline Profile** > **Process Time** graph will sum up the duration of all Functions' processing times. This sum can exceed the Pipeline's total duration displayed in the **Summary**.

Workaround: None identified.

Splunk HEC shows higher outbound data volume than other Splunk Destinations

Reported in	Resolved in	Ticket
3.5.1	TBD	CRIBL-11922

Problem: Events sent to the Splunk HEC Destination will show higher outbound data volume than the same events sent to the Splunk Single Instance or Splunk Load Balanced Destinations, which use the S2S binary protocol.

Workaround: None identified.

Persistent queue with compression requires oversize max queue limit

Reported in	Resolved in	Ticket
3.5.0	TBD	CRIBL-10965

Problem: If you configure persistent queueing to both enable **Compression** and set a **Max queue size** limit, set that limit optimistically and monitor the results. Due to an error in computing the compression factor, Cribl Edge will not fully use **Max queue size** values below or equal to the volume's available disk space. This can lead to a mostly empty disk, lost data (either blocked or dropped), and/or excessive backpressure.

Workaround: With **Compression** enabled, set the **Max queue size** to a value **higher** than the volume's total available physical disk space (disregarding compression). Alternatively, leave **Max queue size** empty, to impose no limit. Monitor overall throughput and the queue size (if PQ engages), to verify that Cribl Edge is maximizing the queue.

GitOps + License expiration = Catch-22

Reported in	Resolved in	Ticket
3.3.1	TBD	CRIBL-8600

Problem: If your Enterprise license expires while you have enabled the GitOps Push workflow, you will encounter the following block: Cribl Stream is in read-only mode, triggering a `Forbidden` error when you try

to update your license key. But you also cannot reset the workflow from `Push` to `None`, because the expired license disables GitOps features.

Workaround: Contact Cribl Support for help updating your license.

Syslog Source's previewed data doesn't proceed through Routes

Reported in	Resolved in	Ticket
3.2.1	TBD	CRIBL-8210

Problem: The Syslog Source's **General Settings** > **Input ID** field's default filter expression is `__inputId.startsWith('syslog:in_syslog:')`. The same filter appears in the **Preview Full** tab's **Entry Point** drop-down, except without the final colon. This means that data sent using **Preview Full**'s version of the filter will not go through Routes that use the **Input ID** version.

Workaround: When sending data from **Preview Full** to a Route, if both use the `syslog:in_syslog` filter, edit the Route's filter to remove the final colon. This will make the filter identical in both places, routing data correctly.

Upgrading a Pack overwrites Lookup and sample-data customizations

Reported in	Resolved in	Ticket
3.0.0	TBD	CRIBL-7998

Problem: Upgrading a Pack overwrites any customizations you've made to the Pack's original/default Lookup tables and sample-data files.

Workaround: Duplicate the Pack's default Lookup tables and sample-data files, and customize your duplicate copies. Upgrading the Pack will not overwrite your local copies.

Re-enabling a Function group mistakenly re-enables all its Functions

Reported in	Resolved in	Ticket
2.1.0	TBD	CRIBL-7053

Problem: When you change a Function group from disabled to enabled, all of its Functions are enabled, regardless of their individual enabled/disabled states when the group was disabled.

Workaround: Avoid disabling and re-enabling Functions as a group (e.g., for testing or stepwise debugging purposes).

Duplicate Workers/Worker GUIDs

Reported in	Resolved in	Ticket
3.0.1	TBD	CRIBL-5611

Problem: Multiple Workers have identical GUIDs. This creates problems in Monitoring, upgrading and versioning, etc., because all Workers show up as one.

Cause: This is caused by configuring one Worker and then copying its `cribl/` directory to other Workers, to quickly bootstrap a deployment.

Workaround: Don't do this! Instead, use the Bootstrap Workers from Leader endpoint.

Can't Enable KMS on Worker Group after installing license

Reported in	Resolved in	Ticket
3.0.0	TBD	CRIBL-5274

Problem: Enabling HashiCorp Vault or AWS KMS on a Worker Group, after installing a LogStream license on the same Group, fails with a spurious `External KMS is prohibited by the current license` error message.

Workaround: On the Leader, navigate to **Settings** > **Worker Processes**. Restart the affected Worker Group's `CONFIG_HELPER` process. Then return to that Worker Group's **Security** > **KMS** Settings, re-enter the same KMS configuration, and save.

Resolved in 4.9

The following issues were resolved in version 4.9 and its maintenance releases.

Windows Event Logs Source (Cribl Edge): imprecise seek position

Reported in	Resolved in	Ticket
4.9.0	4.9.3	CRIBL-29208

Problem: Within the Windows Event Logs Source, a Windows-native module incorrectly interprets the seek position argument as a 32-bit integer, leading to problems seeking to the correct position in the event log. This can lead to duplicate events, longer-than-expected collection times, and other issues.

Workaround: In the Source configuration modal, on the **Advanced Settings** tab, enable **Use Windows Tools**. While this workaround can resolve the seek position problem, it might degrade performance, especially for high-volume logs.

Significant increase in malformed HEC event errors

Reported in	Resolved in	Ticket
4.9.1	4.9.2	CRIBL-29063

Problem: A change to HEC parsing in 4.9.1 might cause an increase in `Dropping malformed HEC event, enable debug to see details` error logs, even when the incoming HEC payloads are known to be valid. Specifically, this can happen in JSON with multi-escaped quote characters (for example, `"type\\\\\\\":\\\\\\\"someType\\\\\\\"`). When this error occurs, Cribl Stream will not ingest the payload and will instead return a `400` error response.

Workaround: None identified. Users are advised to upgrade directly to 4.9.2 or later to avoid this issue.

Worker Nodes or Edge Nodes send false down alerts after turning off load balancing

Reported in	Resolved in	Ticket
4.9.1	4.9.3	CRIBL-28886

Problem: This issue affects Worker Nodes or Edge Nodes with load balancing enabled in a distributed environment. When you disable load balancing, commit and deploy the configuration changes, and restart the

Nodes, Cribl may send a false alert indicating the Nodes are down. Despite the alert, the Nodes remain healthy and operational.

Workaround: None identified.

Unable to upgrade or install Edge to 4.9.0 on Windows machines running .NET version older than 4.7.2

Reported in	Resolved in	Ticket
4.9.0	4.9.1	CRIBL-28774

Problem: Users are not able to install or upgrade to Edge 4.9.0 on Windows machines that are running .NET framework versions older than 4.7.2.

Workaround: Upgrade instances of .NET on your Windows machine to 4.7.2 or newer, then rerun the Cribl Edge installation or upgrade.

REST Collectors query unexpected pages with dual pagination

Reported in	Resolved in	Ticket
4.9.0	4.9.1	CRIBL-28716

Problem: A REST Collector configured to use pagination for both the discovery and collection steps could collect the wrong page. The field used to track the current page for the discover step interferes with the page tracking for the collect step. This causes the collection step to query unexpected pages, leading to inconsistent results.

Workaround: None identified.

Cannot use some words when accessing properties in Pipelines

Reported in	Resolved in	Ticket
4.9.0	4.9.1	CRIBL-28551

Problem: When Pipelines refer to property names containing certain words, an error message occurs: `Failed to build property accessor (...) err=Unallowed ref to '<keyword>'`. The words causing issues are: `AsyncFunction`, `AsyncGenerator`, `AsyncGeneratorFunction`, `Function`, `GeneratorFunction`, `Promise`, `Proxy`, `Reflect`, `Symbol`, `WebAssembly`, `console`, `eval`, `global`, `globalThis`, `process`, `setInterval`, `setTimeout`, `uneval`, `window`.

Workaround: When using any of the above words as property names, surround them by quotation marks, for example: `process.part.part` -> `'process.part.part'`.

Cribl Copilot for Pipelines does not work if accessed from Stream Projects

Reported in	Resolved in	Ticket
4.8.2	4.9.1	AI-802

Problem: After upgrading to Cribl 4.8.2, the "Add function with Copilot" feature does not work when accessed from **Projects** > **Data Projects** > **Pipelines**.

Workaround: Access the "Add function with Copilot" feature in Cribl Stream by going to **Processing** > **Pipelines** > select a Pipeline > **Add Function**.

Google Cloud Pub/Sub Source not populating `attributes` field in events

Reported in	Resolved in	Ticket
4.8.2	4.9.1	CRIBL-28408

Problem: The attributes from a PubSub message do not translate to the Cribl event because the `message.attributes` property from the `PubSub.Message` type is treated as an array instead of an object, resulting in a missing `__attributes` field.

Workaround: None identified.

REST Collector configuration no longer supports the ability to add placeholder values

Reported in	Resolved in	Ticket
4.9.0	4.9.1	CRIBL-28383

Problem: When you configure a REST Collector as a JSON file, you can normally define a placeholder value for various fields. Then, when you switch to the user interface, it prompts you to fill in the placeholders with the actual values. This feature is missing in Cribl Edge 4.9.0.

Workaround: None identified.

Users can't upgrade Cribl in-product when the `CRIBL_VOLUME_DIR` environment variable is set

Reported in	Resolved in	Ticket
4.8.2	4.9.0	CRIBL-28216

Problem: On Cribl Leaders, Cribl Edge, or Cribl Stream Linux instances that have the `CRIBL_VOLUME_DIR` environment variable set, you can't upgrade from 4.8.2 to 4.9.0 using the UI. This affects only upgrades from version 4.8.2.

These scenarios are **not** impacted by this issue:

- Upgrading from versions 4.8.1 and older to 4.9.0.
- Installations where `CRIBL_VOLUME_DIR` is not set.
- Windows installations.
- Containers or RPM installs.

Workaround: Upgrade these instances manually. See Upgrading Cribl Edge on Linux (CLI) for instructions.

The Cloud persistent queue (PQ) health checker sends false alerts for staging environment with only one Worker Node

Reported in	Resolved in	Ticket
4.9.0	4.9.1	CRIBL-28207

Problem: When a staging environment has only one Worker Node, the Cloud PQ health checker creates a critical alert with the message "cloud worker pq config errors".

Workaround: None identified.

`RangeError: offset is out of bounds` and nested field serialization issues in Splunk Load Balanced Destination with S2Sv4

Reported in	Resolved in	Ticket
4.0.0	4.9.0	CRIBL-28080

Problem: The Splunk Load Balanced Destination using S2Sv4 encounters issues when processing certain events:

1. When events contain a field with an array value with many items, parsing could fail with the error message `RangeError: offset is out of bounds.`
2. When events contain a field with a non-stringified object value greater then 1 KB, and **Nested field serialization** is set to `JSON`, the value could be truncated after stringification. The event is still sent but may miss some of the stringified object value.

Workaround:

1. Smaller-sized arrays may help.
2. Serialize nested fields as JSON before they reach the Splunk Destination with the Eval function.

Deploying configurations to Edge Subfleets sometimes fail with TLS enabled

Reported in	Resolved in	Ticket
4.8.2	4.9.0	CRIBL-27793

Problem: In a Cribl-managed instance of Edge, attempting to deploy configuration changes to a Subfleet sometimes fails if the Subfleet, or its parent Fleet, has TLS enabled and contains a certificate with a passphrase.

Workaround: There are two workaround options, in the Fleet or Subfleet settings:

- Use a certificate without a passphrase.
- Disable TLS in **API Server Settings**.

HMAC authentication fails due to empty signature strings

Reported in	Resolved in	Ticket
4.8.0	4.9.0	CRIBL-27682

Problem: HMAC authentication could fail when an API requires empty strings in the signature string builders. Cribl currently filters out any `falsy` values, including empty strings, which can prevent the correct generation of the HMAC value and lead to authentication failures.

Workaround: Varies. See a workaround example in our templates repository.

Persistent queue settings are missing in hybrid Worker Groups

Reported in	Resolved in	Ticket
4.8.1	4.9.0	CRIBL-27644

Problem: On Cribl.Cloud, persistent queue settings are unavailable in the UI for Sources and Destination running in hybrid (customer-managed) Worker Groups.

Workaround: You can configure persistent queues on hybrid Worker Groups by using the **Manage as JSON** option in Sources or Destinations.

A Google Cloud Pub/Sub Source authentication tooltip gives incorrect Google authentication information

Reported in	Resolved in	Ticket
4.8.2	4.9.1	CRIBL-27506

Problem: When you are configuring the Google Cloud Pub/Sub Source, the Authentication screen has a field named **Authentication method** with an incorrect tooltip. Starting in 4.8.2, you should no longer use the environmental variables `PUBSUB_PROJECT` and `PUBSUB_CREDENTIALS` to authenticate with Google, as stated in the tooltip.

Workaround: In 4.8.2 or later, use the `GOOGLE_APPLICATION_CREDENTIALS` environment variable to provide the location of a credential JSON file or the JSON file contents. 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.

C.Time Functions can crash Edge Nodes with old timestamps

Reported in	Resolved in	Ticket
4.7.3	4.9.0	CRIBL-26719

Problem: The `C.Time.s3TimePartition()` and `C.Time.timePartition()` Functions, commonly used for S3 and Cribl Lake Destinations, are susceptible to crashing Worker Processes when encountering events with timestamps prior to January 1st, 1970 (Unix epoch). These events, represented by negative (or `0`) values in the `_time` field, cause Edge Nodes to consume 100% CPU and become unresponsive.

Workaround: To prevent Edge Nodes crashes, insert a Pipeline before any Function or Destination using these methods. Configure the Pipeline to modify the `_time` field of affected events to a positive value (`>0`) (e.g., using the `Date.now()` function).

Unable to Load CPU Profiles in Cribl.Cloud

Reported in	Resolved in	Ticket
4.7.3	4.9.0	CRIBL-26311

Problem: Cribl.Cloud is currently experiencing an issue where users cannot view or download CPU profiles directly through the UI. CPU profiling is useful in troubleshooting performance issues.

Workaround:

1. **Open DevTools**: While on the Worker Process details page, right-click anywhere and select **Inspect**.
2. **Navigate to the Network Tab**: Locate the **Network** tab within DevTools.
3. **Filter Network Requests**: In the filter bar at the top of the Network tab, type `download`. This will filter the requests to only show those related to CPU profile downloads.
4. **Identify API Call**: Look for the latest network request with a path containing `download` (e.g., `/api/v1/workers/<worker_id>/profile/download`). This is the backend API call that fetches the

CPU profile data.

5. **Access Profile Data**: Right-click on the identified API call and select **Open in Sources panel**.
6. **Save Profile**: Copy this data and save it to a file with a `.cpuprofile` extension.

For profile visualization, consider using an open-source tool like Speedscope after saving the profile data. A more permanent solution involving UI updates is planned for a future release.

REST Collector state not updated on job cancellation

Reported in	Resolved in	Ticket
4.7.2	4.9.0	CRIBL-26485

Problem: Canceling a REST Collector job from the Job Inspector before it completes prevents state updates for events collected prior to cancellation. As a result, no state is saved for uncompleted tasks within the collection run.

Workaround: None identified.

When Source PQ is enabled, disk space fills to near capacity and Cribl Edge issues an error

Reported in	Resolved in	Ticket
4.7.2	4.9.0	CRIBL-26478

Problem: With Source-side persistent queues (PQs) enabled, the system is not releasing files correctly, causing the disk space to fill and potential Edge Node failures. In addition, the system frequently logs the following error: `Cannot read properties of undefined (reading 'on')`.

Workaround: You may be able to mitigate this issue by adjusting your Source-side PQ settings. Significantly decreasing the **Max buffer size** and increasing the **Max file size** may resolve the issue.

Chain Functions in Cribl Stream Projects can lead to infinite loops

Reported in	Resolved in	Ticket
4.7.3	4.9.0	CRIBL-25524

Problem: Currently, users can create infinite loops within a project by referencing a Pipeline within itself using a Chain Function.

Workaround: Until a permanent fix is implemented, users should avoid creating circular dependencies within Pipelines within a single project. Do not reference a Pipeline within itself using a Chain Function.

Upgrading to 4.7.0 with Google Cloud Security Operations Destination (formerly Google Chronicle)

Reported in	Resolved in	Ticket
4.7.0	4.9.0	CRIBL-25371

Problem: When upgrading to Cribl Edge 4.7.0, if an existing **Google Chronicle Destination** (now Google Cloud Security Operations) is configured on a particular Fleet, all Destinations in that Fleet will fail to load and process data until a commit and deploy is performed on the Fleet.

If you use the **Google Chronicle Destination**:

Recommended: Proceed with the upgrade, but be prepared to immediately perform a commit and deploy on affected Fleets after upgrading Edge Nodes and Leaders. Edge Nodes will only resume data processing after a successful commit and deploy.

Workaround: None identified.

Licensing page displays a permission access warning when a Cloud Connection is enabled

Reported in	Resolved in	Ticket
4.7.0	4.9.0	CRIBL-25147

Problem: When you have an on-premises Cloud Connection to Cribl.Cloud and navigate to the Licensing page, a warning displays that the user does not have permission to access the resource. You won't be able to access the license page with a connected environment (as your Cribl.Cloud account now handles credits for the connected environment) but it is not a permissions issue.

Workaround: To view billing information for your connected environments, log into your Cribl.Cloud account and go to **Organization**, then **Billing & Usage**.

OTLP Metrics Function does not process metrics unless all metric names are quoted literals

Reported in	Resolved in	Ticket
4.6.1	4.9.0	CRIBL-24877

Problem: When the Publish Metrics Function is followed by the OTLP Metrics Function in a Pipeline, the OTLP Metrics Function may fail to produce any results. This failure is possible when the metric name expression for any of the published metrics uses an unquoted string literal and/or evaluates to a numeric value.

Workaround: If you must use a metric name expression in Publish Metrics, wrap the results of the expression in single or double quotes to ensure that it returns a string.

Optionally, leave the metric name expression blank and use the Rename Function to rename the metric names before sending them to the OTLP Metrics Function.

A File Monitor memory leak causes high memory usage only on Windows Edge Nodes

Reported in	Resolved in	Ticket
4.4.4	4.9.1	CRIBL-22970

Problem: When Cribl Edge is deployed on a Windows Node and the File Monitor Source is configured on that Node, a memory leak will cause high memory usage on the Windows Edge Node.

Workaround: None identified.

Kafka and Confluent mTLS requirement limits Schema Registry authentication

Reported in	Resolved in	Ticket
1.2.0	4.9.0	CRIBL-21047

Problem: The Kafka and Confluent Cloud Source and Destination only support mutual TLS (mTLS) authentication for the Kafka Schema Registry. This prevents you from configuring separate authentication

mechanisms for the Schema Registry and Kafka brokers, which is necessary when the Schema Registry and brokers are managed by different departments with distinct authentication requirements.

Workaround: None identified.

Healthy Destinations spuriously report `Blocked Status`

Reported in	Resolved in	Ticket
4.2.1	4.9.0	CRIBL-19322

Problem: Healthy TCP-based, Kafka-based, Splunk, and Metrics Destinations can spuriously report `Blocked Status` in the **Charts** tab, sometimes also logging the blocked status inaccurately.

Workaround: None identified.

Resolved in 4.8

The following issues were resolved in version 4.8 and its maintenance releases.

Removing Member from Team removes access after a lag

Reported in	Resolved in	Ticket
4.8.2	4.8.2+	CRIBL-27680

Problem: As the Owner or Admin of a Cribl.Cloud Organization, if you remove a Member from a Team, that user will retain any Team-associated Permissions on products and product resources for up to 30 minutes. (This lag does not affect Team changes on on-prem Organizations.)

Workaround: The intended access changes will take effect within 30 minutes.

In Cribl Edge, TLS certificate paths in Sources and Destinations are incorrect

Reported in	Resolved in	Ticket
4.8.0	4.8.2	CRIBL-26971

Problem: After Cribl Edge upgrades to 4.8.0 or later, TLS certificate paths for Cribl Edge Sources and Destinations are not updated. This may affect data flow in Fleets and Subfleets containing TLS-configured Sources or Destinations.

Workaround: Update the TLS certificate path.

1. In the parent Fleet, navigate to the Source or Destination that uses a TLS certificate.
2. In the **Certificate path** field, change `default` to `local`.
   For example:
   - Old path: `groups/default_fleet/default/edge/auth/certs/asd.crt`
   - New path: `groups/default_fleet/local/edge/auth/certs/asd.crt`
3. Save the Source or Destination.
4. Select **Commit and Deploy with Subfleets** so the Subfleets receive the change.

Splunk Destinations incorrectly detect that a downstream Splunk instance is shutting down

Reported in	Resolved in	Ticket
4.8.0	4.8.1	CRIBL-26969

Problem: Splunk Single Instance and Splunk Load Balanced Destinations might incorrectly detect that a downstream instance of Splunk is shutting down at frequent intervals. This only affects Destinations with **Max S2S version** set to `v3` and the **Minimize in-flight data loss** toggle enabled.

Workaround: Change the **Max S2S version** setting for the Destination to `v4`. Or, disable the **Minimize in-flight data loss** toggle.

Incoming data from Collectors sometimes incorrectly displays UTF-8 characters

Reported in	Resolved in	Ticket
4.8.0	4.8.2	CRIBL-26967

Problem: Cribl Stream sometimes displays incoming data containing multi-byte UTF-8 characters incorrectly when Collectors such as AWS S3 ingest events.

Workaround: None identified.

Wiz Source does not correctly paginate

Reported in	Resolved in	Ticket
4.7.2	4.8.2	CRIBL-26965

Problem: The Wiz Source in Cribl Stream does not correctly handle pagination. The default queries for each data type in the Wiz Source incorrectly pass the `endCursor` parameter, resulting in the same page of data being returned repeatedly.

Workaround: Modify each default query by changing `endCursor: endCursor` to `after: endCursor`.

Some elements are missing from JSON-formatted events collected by the Windows Event Logs Source

Reported in	Resolved in	Ticket
4.8.0	4.8.2	CRIBL-26960

Problem: JSON-formatted events collected by the Windows Event Logs Source don't contain the elements `UserData`, `DebugData`, `BinaryEventData`, and `ProcessingErrorData`.

Workarounds:

You can either:

- Set **Use Windows Tools** to `Yes` in the Windows Event Logs Source Advanced Settings.

Or:

- Set the **Event format** to **XML**. Create a Pipeline that contains the `C.Text.parseWinEvent(_raw)` expression, which will analyze the incoming XML event data and generate JSON-formatted output.

Missing in-product documentation for ServiceNow Cloud Observability Destination

Reported in	Resolved in	Ticket
4.8.0	4.8.2	CRIBL-26958, SAAS-10932

Problem: The ServiceNow Cloud Observability Destination documentation is not available in-product. Attempting to access the documentation within the product interface returns the error "Unable to load docs. Please check Cribl's online documentation instead."

Workaround: Refer to the online documentation for configuration instructions.

Windows Event Log events are mishandled when they're missing provider metadata

Reported in	Resolved in	Ticket
4.8.0	4.8.2	CRIBL-26944

Problem: Events without a valid provider are missing the `_raw` property. This happens in the Windows Event Logs Source when **Use Windows Tools** is set to `No` in Advanced Settings, and Render event message strings is enabled.

Workaround: Set **Use Windows Tools** to `Yes` or disable **Render event message strings**.

`_raw` field expansion is incomplete

Reported in	Resolved in	Ticket
4.7.2	4.8.0	CRIBL-26377

Problem: When capturing live data, the `_raw` field unexpectedly expands only to the first level. It should fully expand to reveal its entire nested JSON structure when tapped.

Workaround: None identified.

Splunk TCP Source and Destination: Incorrect Max S2S Version displayed in UI

Reported in	Resolved in	Ticket
4.6.1	4.8.2	CRIBL-26333

Problem: If a Splunk TCP Source or Destination was created before the **Max S2S Version** setting existed, the UI incorrectly displays `v4` as the **Max S2S Version**, instead of the actual default `v3`. This discrepancy can lead to version mismatch errors when forwarders upgraded to Splunk 9.1.4 attempt to send data.

Workaround: Manually change the **Max S2S Version** dropdown to `v3` and then back to `v4` in the UI, and save the configuration. This ensures the correct version is set in the config.

Unable to Send Data via S2S to Splunk Cloud and Splunk Enterprise

Reported in	Resolved in	Ticket
4.5.0	4.8.0	CRIBL-25842

Problem: Attempting to send data to recent versions of Splunk Cloud or Splunk Enterprise via Cribl Stream's Splunk Single Instance or Splunk Load Balanced Destinations integrations may encounter issues after upgrading Splunk.

Workaround: Use the Splunk HEC Destination to send traffic to Splunk. HEC is the recommended protocol for sending data to Splunk due to its better load balancing, compression, and overall performance.

Jobs stuck in "pending" status

Reported in	Resolved in	Ticket
4.7.2	4.8.0	CRIBL-24336, CRIBL-26359

Problem: Jobs can get stuck in **Pending** state when you commit and deploy configuration while the job is running. This prevents the job from running again, and if multiple jobs freeze, the job queue can fill up. Note that Worker upgrade jobs may fail when upgrading from versions earlier than 4.8.

Workaround: Restart the affected Worker and the Leader. To restart the Worker:

1. Log into Cribl Stream as an Admin.
2. Go to the **Workers** page.
3. Select the affected Worker's GUID. You will teleport to this Worker's UI.
4. On the colored top bar, select **Restart Stream**.

To restart the Cribl.Cloud Leader:

1. Get an API token for your Leader Node.
2. Use the following curl command to restart the Leader:
   ```
   curl "https://<workspace-org>.cribl.cloud/api/v1/system/settings/restart" -H "Authorization: Bearer $token" -H "Content-Type: application/json" --data-raw "{}"
   ```

where `workspace-org` comes from your Cribl.Cloud URL, and `$token` is the API token.

Incorrect Source URL in REST Collector events

Reported in	Resolved in	Ticket
2.4.4	4.8.2	CRIBL-24280

Problem: The REST Collector incorrectly displays the **Discovery URL** as the `source` field in collected events, instead of the actual **Collect URL**. This issue occurs when using OAuth2/token API for authentication.

Workaround: Manually adjust the `source` field in downstream processing or use a custom script to update the field after data collection.

Stream Projects: Post-processing Pipelines defined in Destination do not work

Reported in	Resolved in	Ticket
4.5.0	4.8.0	CRIBL-23197

Problem: When using Cribl Stream Projects, a post-processing Pipeline defined only in the Destination, not in the Project itself, will be ignored.

Workaround: Copy the post-processing Pipeline inside the Project:

1. In Cribl Stream's **Processing** > **Pipelines** table, click the Pipeline you're using for post-processing.
2. Select the Pipeline's gear ( ) button, and then the "Manage as JSON" button at the top right of the Pipeline settings.
3. Select **Export** and save the Pipeline in JSON format in a convenient location.
4. Go back to the **Project** screen and select **Pipeline** at the top.
5. Select **Add Pipeline**, and then **Import from File**.
6. Choose your exported configuration and confirm.

Post-processing should now function correctly in your Project.

Already-existing Webhook Destinations show unexpected UI behavior upon upgrade to 4.4.4

Reported in	Resolved in	Ticket
4.4.4	4.8.2	CRIBL-22184

Problem: If you have a Webhook Destination in a Cribl Edge deployment, and then upgrade that deployment to version 4.4.4, the UI will show an error; the new **Load balancing** toggle will be turned on with associated settings displayed; and the UI will not allow you to save changes.

Workaround: First, toggle **Load balancing** off. This will return the UI to normal behavior. Then, if you want to use the Destination without load balancing, configure as usual and save. Or, if you prefer, update the **Webhook URLs** to include one or more valid endpoints you want to send to, and save the Destination with **Load balancing** enabled.

Resolved in 4.7

The following issues were resolved in version 4.7 and its maintenance releases.

Splunk HEC Destination drops events when `_raw` is an empty string

Reported in	Resolved in	Ticket
1.6.0	4.7.2	CRIBL-25928

Problem: When sending events to a Splunk HEC Destination, if the `_raw` field is an empty string, Cribl Stream sends an empty string as `_raw` instead of trying to send the full serialized event. This results in an error from Splunk and the event is dropped.

Workarounds:

- Do not set `_raw` to an empty string in any Pipelines feeding a Splunk HEC Destination.
- If you do not know or cannot change the `_raw` contents prior to the Destination, add an Eval Function to a post-processing Pipeline with the expression: `_raw = _raw?.length ? _raw : undefined;`.

Splunk HEC Destination drops events with null `_time` values

Reported in	Resolved in	Ticket
1.1.0	4.7.2	CRIBL-25927

Problem: Events sent to a Splunk HEC Destination with a null `_time` field are dropped by Splunk, halting further processing of the entire payload (potential data loss).

Workaround: Assign the `_time` field to the current time in a post-processing Pipeline by adding an Eval Function before the Splunk HEC Destination with the expression: `_time === null ? Date.now() / 1000 : _time`.

Group and Fleet Naming Conflict

Reported in	Resolved in	Ticket
3.3.0	4.7.2	CRIBL-25888

Problem: Groups and Fleets in Cribl Stream can have similar names, potentially causing issues when differentiated only by a hyphen (-) and underscore (_). For example: When configuring a Worker Group (e.g., CRIBL_GOAT) through the user interface (UI), the configuration helper might mistakenly retrieve settings from the similarly named Group or Fleet (CRIBL-GOAT). This crossover can cause significant configuration errors and operational disruptions.

Workaround: To avoid this issue, we recommend adopting a naming convention with distinct names for each Group/Fleet.

REST Collector job state issues with pagination

Reported in	Resolved in	Ticket
4.6.0	4.7.2	CRIBL-25763

Problem: The REST Collector's job state is incorrectly updated between tasks when using pagination. This can lead to inconsistent state tracking and potential data gaps or overlaps in the collected data. You may notice that the state from one paginated task affects subsequent tasks within the same job, causing unexpected behavior.

Workaround: Do not use pagination in your collection jobs alongside state tracking. Instead, schedule individual collection jobs for each page to ensure state isolation.

Unlinked jemalloc library may cause memory spikes

Reported in	Resolved in	Ticket
4.7.0	4.7.2	CRIBL-25702

Problem: jemalloc library being unlinked from the Cribl binary can potentially lead to memory spikes, affecting the stability and performance of the application on large-scale deployments. This issue only affects on-prem deployments using x84_64 architecture.

Workaround: In `/etc/systemd/system/cribl.service`, manually link jemalloc by adding `Environment=LD_PRELOAD=/opt/cribl/bin/libjemalloc.so`:

```
# existing lines
Environment=CRIBL_SERVICE=1
Environment=CRIBL_SERVICE_NAME=cribl
# link the jemalloc library
Environment=LD_PRELOAD=/opt/cribl/bin/libjemalloc.so
```

Syslog Sources with octet-count framing malfunction on unframed TCP messages

Reported in	Resolved in	Ticket
4.0.0	4.7.2	CRIBL-25572

Problem: Syslog Sources configured with octet-count framing (pre 4.7.0), or with `octetCounting: true` (via **Manage as JSON** since 4.7.0), can malfunction if they receive unframed messages over TCP. This will cause the Source to return many warning messages like `Invalid octet count: undefined. Trying to skip to next frame` and the Worker may run out of memory.

Workaround: Do not send non-octet-counted messages to a Syslog Source that is configured for octet-count framing.

Project Editor unable to edit Pipelines

Reported in	Resolved in	Ticket
4.7.1	4.7.2	CRIBL-25559

Problem: A user assigned the `Editor` permission within a Cribl Stream project is unable to edit pipelines in that project. When attempting to edit a **Pipeline**, the user encounters a permissions error.

Workaround: None identified.

Collector and Source state lost during Leader failover

Reported in	Resolved in	Ticket
4.6.0	4.7.2	CRIBL-25539

Problem: When a Leader Node fails over, the state for REST Collectors, Database Collectors, Wiz Sources, Kinesis Sources, and Windows Event Forwarder (WEF) Sources is not preserved.

Workaround: None identified.

Prometheus summary metrics in OTLP Format cause Kafka Destination errors

Reported in	Resolved in	Ticket
4.7.0	4.7.2	CRIBL-25435

Problem: Using a Prometheus Source for summary metrics, the `quantile_values` are formatted as an object. However, for OTLP (OpenTelemetry Protocol) serialization, `quantile_values` must be in array format. This discrepancy causes errors when attempting to write summary metrics to Kafka Destinations, resulting in failed metric transmissions.

Workaround: None identified.

SQL Server Collector Database Connection fails with Config authentication

Reported in	Resolved in	Ticket
4.7.0	4.7.2	CRIBL-25418

Problem: When upgrading to Cribl Stream 4.7.0, SQL Server Collectors configured to use the `Config` authentication method will fail to load, causing the Database Collector to stop working. The only valid authentication method for SQL Server Collectors will be the `Connection String`. Additionally, the section to manage Database Connections (under Knowledge) will not load existing connections.

Workaround: Manually edit each `database-connections.yml` configuration file where SQL Server Database Connections use the `configObj` authentication type. Change the authentication method to

`connectionString`.

1. To convert a config object to a connection string, refer to the relevant fields in this table: SQLConnection.ConnectionString. For each line in the config object, take the key, find the corresponding label in the table, and create a key-value pair in the connection string. For example, if the config object is `{ userid: 'test', server: 'tcp:servername portnumber' }`, the connection string should be: `userID=test;server=tcp:servername portnumber`.
2. Edit the `database-connections.yml` file, change `authType` from `configObject` to `connectionString`, and add the connection string to the file. The config object can remain, but a valid connection string must be added to get the Database Collectors working again in Cribl Stream.
3. To store the connection string encrypted you need to save the configuration in the UI. Make a minor change to the Database Connection, like, add and then remove a character in the description, and then save the configuration. Commit and deploy the changes as necessary.

Extra Auth fields are visible when configuring a Splunk Load Balanced Destination

Reported in	Resolved in	Ticket
4.7.0	4.7.2	CRIBL-25411

Problem: When configuring a Splunk Load Balanced Destination, you will see an extra **Authentication method** and **Auth token** field in **General Settings** (the fields are visible just above **OPTIONAL SETTINGS** and are outside of the **Authentication tokens** group). These fields are not needed, ignore them.

Workaround: Disregard the extra fields and use the fields in the **Authentication tokens** group instead.

Syslog source UDP events incorrectly parsed when not starting with < character

Reported in	Resolved in	Ticket
4.7.0	4.7.1	CRIBL-25400

Problem: When sending events via UDP to a Syslog Source where the events are not valid syslog and do not start with a < character, events may be broken incorrectly, leading to warnings, errors, and incorrectly parsed events.

Workarounds:

- Use the **Manage as JSON** option on the Source config and add this line: `inferFraming: false`.
- Use the Raw UDP Source instead for UDP datagrams that are not valid syslog.

Some Sources and Destinations don't appear on the Health page for a Fleet

Reported in	Resolved in	Ticket
4.7.0	4.7.2	CRIBL-25372

Issue: Some Sources and Destinations will not appear on the **Health** page for a Fleet. This appears to affect predominantly Windows Sources.

Workaround: Navigate to the specific Source or Destination and select **Chart** to view Fleet health metrics.

When upgrading from the Leader, some Windows Edge Nodes can fail to upgrade

Reported in	Resolved in	Ticket
4.6.0	4.7.2	CRIBL-25339

Problem: When upgrading from the Leader, some Windows Edge Nodes fail to upgrade. This occurs intermittently and is not predictable. This issue may also cause the Windows Edge Node to lose its connection to Leader and stop sending data.

Workaround: If you encounter this issue, you can manually uninstall and reinstall Cribl Edge on any failed Windows Nodes. However, Cribl Edge Nodes on version 4.6.0 continue to be compatible with Leader versions between versions 4.6.0 and 4.7.0 so you do not need to upgrade Edge Nodes at this time. Specifically, we recommend that you do not set a `Target Version` on Fleets that contain Windows Edge Nodes until we resolve this issue.

REST Collectors fail to capture response headers with pagination

Reported in	Resolved in	Ticket
4.6.1	4.7.2	CRIBL-25018

Problem: When you use REST Collectors with pagination enabled and turn on the **Capture response headers** toggle (introduced in the 4.6.1 release and off by default), the response headers are not captured as expected. When you toggle on **Show internal fields** in the result and expand the `__collectible` object, the `resHeaders` field is missing. This issue prevents you from accessing response header information during paginated data collection.

Workaround: None identified.

Unable to upgrade Leader and Workers from the UI when using CDN

Reported in	Resolved in	Ticket
4.6.0	4.7.0	CRIBL-24999

Problem: You're unable to upgrade the Leader and Worker Nodes from the UI. The upgrade process fails to check compatibility for the CDN, causing the upgrade to be blocked. The UI may incorrectly indicate that the currently installed version is the latest, even when newer versions are available in the CDN.

Workaround: Point the upgrade process to the download **Path** instead of the CDN.

Incorrect `bucket_counts` format in OTLP Metrics via Kafka Destination

Reported in	Resolved in	Ticket
4.5.0	4.7.0	CRIBL-24860

Problem: When sending Prometheus histogram metrics to an OTLP-compatible Destination via Kafka, the `bucket_counts` field is incorrectly formatted as an object instead of an array. This causes errors during Protobuf encoding, specifically the error: `.opentelemetry.proto.metrics.v1.HistogramDataPoint.bucket_counts: array expected`. This results in failed metric transmissions and potential data loss.

Workaround: None identified.

Full Preview doesn't work with Packs used as pre-processing Pipeline

Reported in	Resolved in	Ticket
4.5.1	4.7.2	CRIBL-24516

Problem: Pipeline's **Full Preview** tab presents incorrect information when using a pre-processing Pipeline coming from a Pack. This occurs only when previewing from the Leader – **Full Preview** from a Edge Node displays data correctly.

Workaround: Teleport to a Edge Node and run **Full Preview** from the Edge Node level.

Throttling causes some Destination senders to reconnect at each DNS resolution interval

Reported in	Resolved in	Ticket
4.5.1	4.7.0	CRIBL-24505

Problem: When you've configured **Throttling** on a Destination, a design conflict with the DNS resolution process leads to the reconnection of all senders in each DNS resolution cycle (every 10 minutes by default). This could cause the Destination to temporarily block and/or engage PQ, depending on the configuration, until senders reconnect. This affects the Cribl TCP, Syslog, and TCP JSON Destinations when load balancing is enabled.

Workaround: Don't use **Throttling** at the Destination.

Inactive Destinations in GitOps environments block data flow

Reported in	Resolved in	Ticket
4.4.4	4.7.2	CRIBL-22623

Problem: In GitOps environments, inactive Destinations are causing data flow to be blocked because the default backpressure behavior is to block. When a Destination is inactive, it triggers backpressure, preventing data from being processed and sent to other active Destinations.

Workaround: Change the backpressure behavior from `Block` to `Drop` or `Buffer` for the affected Destinations.

Configuration updates not shared between primary and standby Leaders

Reported in	Resolved in	Ticket
3.5.0	4.7.0	CRIBL-12814, CRIBL-14556

Problem: With high availability/failover enabled, updates to the active Leader's configuration are not automatically synced to the standby Leader. Workers might not be able to connect to the standby Leader when it takes over.

Workaround: Use the filesystem to explicitly sync the updated `instance.yml` file across the two Leaders' hosts. In 4.0.3 through 4.6.1, as an intermediate fix, enabling HA will prevent further UI-based changes to **Distributed Settings**. This will enforce config changes via edits to portable `instance.yml` files.

GitOps Push mode doesn't support ad hoc Collection jobs

Reported in	Resolved in	Ticket
3.5.3	4.7.2	CRIBL-12868

Problem: If you've enabled the GitOps Push workflow, you will be unable to run ad hoc Collection jobs.

Workaround: There are two options. 1. Temporarily disable the Push workflow in your environment. 2. Create a scheduled Collection job (with a relaxed cron schedule) on your dev branch, and push it to production through your Push workflow.

Resolved in 4.6

The following issues were resolved in version 4.6 and its maintenance releases.

Some licensing limits are being applied incorrectly

Reported in	Resolved in	Ticket
4.6.0	4.6.1	CRIBL-24687

Problem: After the Leader restarts, Cribl Edge may not apply your license correctly. Because of this error, you may have insufficient user privileges for certain features and limits. For example, Cribl Edge may not correctly

apply certain licensing limits, such as the total data volume allowed. Under certain conditions, this issue can block data ingestion. Note that this issue is timing-dependent so it can affect different areas of the product after a Leader restart.

Workaround: Two workarounds are available:

- Reapply the license in the user interface without restarting the Leader.
- Wait approximately one hour for the license manager to perform a periodic license check. This will reapply the license and may resolve the issue.

In environments that use a proxy or a TLS-breaking security appliance, the Worker / Edge Node to Leader communication is disrupted

Reported in	Resolved in	Ticket
4.6.0	4.6.1	CRIBL-24451

Problem: In version 4.6.0, we added SNI routing to Cribl Edge and Cribl Stream. This caused a communication issue between the Leader Node and Worker / Edge Nodes in proxied deployments of Cribl Stream and Edge. Edge and Worker Nodes reference the Leader Node using the TLS server name, `stream` or `edge`, which web proxies will block.

Workaround: There are several workaround options:

- Disable SNI routing for firewalls / web proxies. See our instructions here: Disable SNI Routing.
- Downgrade the Stream Worker / Edge Node to `4.5.1`.
- Bypass Worker / Edge node traffic to the Crib.Cloud Leader or local Leader around the proxy or security appliance.
- (Applies to certain security appliances only) Verify whether connections to the hostname `stream` or `edge` are being blocked on the security appliance, and add an exception or bypass.
    - Alternatively, modify the `/etc/hosts` file on Worker / Edge Nodes (`C:\Windows\system32\drivers\etc\hosts` for Windows Edge Nodes) to override the hostname(s) `stream` / `edge` to `127.0.0.1`. This prevents false-positive threat evaluations on the security appliance from blocking Worker / Edge to Leader traffic, and potentially block inputs/outputs on Worker / Edge Nodes.

Cribl Edge displays a permissions-related error in some areas of the user interface

Reported in	Resolved in	Ticket
4.6.0	4.6.1	CRIBL-24310

Problem: This issue is another form of CRIBL-24687, documented above.

Workaround: See CRIBL-24687.

Limits settings for Redis are not available in Cribl.Cloud

Reported in	Resolved in	Ticket
4.6.0	4.6.1	CRIBL-24269

Problem: The **Limits** settings at Fleet level that on-prem deployments use are not available in Cribl.Cloud. Because the Redis settings (including **Reuse Redis connections**) are within that **Limits** UI, Edge Nodes in Cribl.Cloud do not have access to those settings.

Workaround: None identified.

UI can omit details when viewing Email Notifications via Manage > Notifications menu

Reported in	Resolved in	Ticket
4.6.0	4.6.1	CRIBL-24169

Problem: When an Organization Admin views an Email Notification via the **Manage** > **Notifications** menu, the notification's condition and/or other details may appear empty even though they exist. This also affects Product Admins for on-prem deployments.

Workaround: The Organization or Product Admin should view the notification this way instead: In a Worker Group, open the **Notifications** tab for the Source that was configured to send the notification. The notification and all its details will appear as usual.

Data Lake Amazon S3 Destination partitioning scheme is incorrect

Reported in	Resolved in	Ticket
4.5.1	4.6.1	CRIBL-23878

Problem: When you create a new Data Lake Amazon S3 Destination, its partitioning scheme will be incorrect.

Workaround: Deleting the incorrect partitioning scheme will cause the Destination to replace it with the correct one. To do this:

1. Open the Destination's configuration modal and click **Manage as JSON**.
2. Find the row beginning with the key `"partitionExpr"` – its value will be the incorrect partitioning expression `C.Time.strftime(_time ? _time : Date.now()/1000, '%Y/%m/%d')`.
3. Delete that entire row.
4. Click **OK** and save the changes.
5. To verify, reopen the configuration modal and click **Manage as JSON**. Check the `"partitionExpr"` row – its value will now be the correct partitioning expression `C.Time.s3TimePartition(_time ? _time : Date.now()/1000, 'h')`.

Splunk Single Instance and Splunk Load Balanced Destinations sending high-cardinality events over S2S v4 can leak memory

Reported in	Resolved in	Ticket
4.0.0	4.6.1	CRIBL-23793

Problem: The memory leak can happen when **Advanced Settings > Max S2S version** is set to `v4` and the combinations of `source`, `sourcetype`, and `host` on events flowing out of the Destination exceed 300 per Worker Process.

Workaround: Set **Advanced Settings > Max connections** to a non-zero value. This reduces the number of senders active at any one time, while distributing load more evenly as Cribl Edge rotates randomly through downstream IPs once per load balancing period.

Leader incorrectly indicates version mismatch for Edge Nodes

Reported in	Resolved in	Ticket
4.5.0	4.6.1	CRIBL-23657

Problem: After upgrading an instance of Cribl Edge with `CRIBL_VOLUME_DIR` set, if the Leader has connected Edge Nodes running different versions, the warning icon is incorrectly located next to Edge Nodes running the same version as the Leader, instead of those running the Leader's prior version.

Workaround: Manually restart Cribl Edge after upgrading with `CRIBL_VOLUME_DIR` set.

Permissions error appears when you try to edit a Pipeline

Reported in	Resolved in	Ticket
4.5.1	4.6.0	CRIBL-23627

Problem: Cribl Stream Project Members with the Editor permission cannot view or edit Pipelines within the Stream Project.

Workaround: None identified.

An error can occur when you click Save to create a new Worker Group or Fleet

Reported in	Resolved in	Ticket
4.5.1	4.6.0	CRIBL-23496

Problem: In Edge and Stream, an error can occur when you click **Save** to create a new Worker Group or Fleet. This error appears as a warning: `The Config Helper service is not available because a configuration file doesn't exist or the settings are invalid. Please fix it and restart Cribl server.`

Workaround: Refresh the page.

Amazon S3 Source and Destination, Amazon SQS Destination, and CrowdStrike Source crash in FIPS mode

Reported in	Resolved in	Ticket
4.5.1	4.6.0	CRIBL-23436

Problem: When Cribl Edge in FIPS mode attempts to read or write files, the Amazon S3 Source and Destination, Amazon SQS Destination, and CrowdStrike Source crash with a `digital envelope routines::unsupported` error. This happens because the underlying Amazon AWS SDK v2 is configured to use the MD5 algorithm, which is not allowed in FIPS mode. Cribl Stream 4.6.0 fixes this problem by configuring the AWS SDK to skip the checksum computation that used MD5.

Workaround: None identified.

Extracted traces from the OpenTelemetry Source not sent to OpenTelemetry Destinations

Reported in	Resolved in	Ticket
3.2.0	4.6.1	CRIBL-23394

Problem: When the OpenTelemetry Source is configured to **Extract spans**, the resulting span event cannot be passed to OpenTelemetry Destinations, like Honeycomb.

Workaround: Disabling **Extract spans** allows batched spans to be ingested without problems.

Diagnostic bundles cannot be created when git is enabled

Reported in	Resolved in	Ticket
4.4.0	4.6.0	CRIBL-23374

Problem: Attempting to create a diagnostic bundle fails when git is enabled, resulting in an error.

Workaround: In the **Create & Export Diag Bundle** window, disable **Include git**.

Diag doesn't collect logs when the `log` directory is a link

Reported in	Resolved in	Ticket
4.5.0	4.6.0	CRIBL-23312

Problem: When the `log` directory is a symbolic link, diag doesn't collect log information (without throwing any errors).

Workaround: Avoid using symbolic links for the `log` directory, or use a direct mount.

Users cannot log into unsecured LDAP servers

Reported in	Resolved in	Ticket
4.5.0	4.6.0	CRIBL-23237

Problem:

With **Access Management** > **Authentication** > **Type** set to `LDAP` and **Secure** toggled to `No`, you should be able to configure and run an unsecured (`ldap://`) LDAP server. See LDAP Authentication. However, this bug causes the **Secure** toggle to have no effect when set to `No`, meaning that your LDAP server must be configured with secure (`ldaps://`) connections or you will not be able to log in.

Workaround: There are two possible workarounds:

1. Run your LDAP server with secure LDAP authentication.
2. If you wish to run your LDAP server **without** secure LDAP authentication, downgrade or upgrade to run a Cribl Edge version **other than** 4.5.0 or 4.5.1.

Azure Data Explorer (ADX) Destination does not retry HTTP requests that come back with `520` response codes

Reported in	Resolved in	Ticket
4.4.4	4.6.1	CRIBL-23189

Problem: Cribl Edge should, but does not, retry HTTP requests that return an `520 Internal server error` response code.

Workaround: None identified.

Windows upgrades of Cribl Edge do not respect original installation directory

Reported in	Resolved in	Ticket
4.4.4	4.6.1	CRIBL-23032

Problem: When upgrading a Windows installation of Cribl Edge that was originally installed in a custom Windows directory (`E:\` for example), the upgrader moves Cribl Edge back to the `C:\` directory.

Workaround: None identified.

Product Documentation: Reference Architecture image download link is broken

Reported in	Resolved in	Ticket
4.1.0	4.6.1	CRIBL-22502

Problem: The editable diagram that appears in the All-in-One Reference Architecture section of the documentation is broken for all non-current release versions. Clicking the link for the `SVG` or `draw.io` file download results in a 403 error.

Workaround: Use the links in the most recent version of the documentation.

Splunk TCP Source logs include an unclear message about an unsupported payload

Reported in	Resolved in	Ticket
4.0.0	4.6.0	CRIBL-21845

Problem: Because the Splunk TCP Source does not support ingesting compressed data via the Splunk S2S protocol, it cannot parse such payloads, and a correct error message in this situation would say “Could not parse payload. Turn compression off in upstream sender and try again.” Instead, the Source logs the unclear message “Failed to parse S2S payload”.

Workaround: None identified.

Resolved in 4.5

The following issues were resolved in version 4.5 and its maintenance releases.

High-volume UDP data dropped in Cribl.Cloud

Reported in	Resolved in	Ticket
4.1.3	4.5.0	SAAS-4399

Problem: Ingesting high rates of UDP events per second can cause Cribl.Cloud to drop some of the data. This limitation of the UDP protocol affects UDP-supporting Sources: Raw UDP, Metrics, SNMP Trap, and Syslog in UDP mode.

Workaround: To minimize the risk of data loss, deploy a hybrid Stream Worker Group, with Worker Nodes as close to the UDP senders as possible.

Cribl also recommends tuning the OS UDP buffer size.

S3 Collector Path extractors setting has no effect

Reported in	Resolved in	Ticket
4.4.4	4.5.1	CRIBL-23338

Problem: In the S3 Collector, the **Collector Settings** > **Path extractors** setting has no effect because of a defect in the `index.js` file.

Workaround: Contact Cribl Support for assistance replacing the `index.js` file with a corrected one.

Running a reverse proxy with a custom baseUrl breaks the user interface

Reported in	Resolved in	Ticket
4.5.0	4.5.1	CRIBL-23089

Problem: If Cribl Edge is configured with a custom baseUrl, Leader deployments sitting behind a reverse proxy server experience user interface errors. The user interface fails to load data correctly from the backend, URLs fail to display the custom baseUrl address in the address bar, and using an address with the custom baseURL results in a 404 error.

Workaround: None identified.

The API Reference automatic authorization doesn't work in Cribl.Cloud

Reported in	Resolved in	Ticket
4.5.0	4.5.1	CRIBL-22822

Problem: In Cribl.Cloud, the API Reference (under **Global Settings**) uses an invalid access token. Because of this, you will receive an HTTP 401 error when you attempt to run an API command.

Workaround: To obtain a valid token, do the following:

1. Open the Developer Tools network tab in your browser and copy the authorization bearer token from one of your requests. Alternatively, you can follow the docs steps here to create a Cribl.Cloud API credential and obtain a token. Copy the token.
2. On the API Reference page, click **Authorize**.
3. In the resulting modal, click **Logout**.
4. Enter your copied token in the **Value** field and click **Authorize**.

When in the context of a Pack, the Pipeline link in the routing table is broken

Reported in	Resolved in	Ticket
4.5.0	4.5.1	CRIBL-22762

Problem: In the Pack context, under **Routes**, if you click the hyperlinked Pipeline name in the table, you will get an `Item not found` error. Clicking `Try again` from the error page does not resolve the issue. However, the link icon at the end of the Pipeline field in the routing table works as expected and brings you to the intended Pipeline page.

Workaround: Use the link icon at the end of the Pipeline field in the Route to access the Pipeline.

OTLP Metrics function does not parse `__criblMetrics` when metric values are strings

Reported in	Resolved in	Ticket
4.5.0	4.5.1	CRIBL-22731

Problem: Documentation for the Publish Metrics Function indicates that it evaluates the **Metric Name Expression** to the **Event Field Name**, but it actually returns a `null`. As a result, the OTLP Metrics function is unable to properly parse `__criblMetrics` when the metrics values are strings.

Workaround: When using Publish Metrics and OTLP Metrics together, use the Numerify Function before the OTLP Metrics Function.

Fleet-level log searches no longer work

Reported in	Resolved in	Ticket
4.5.0	4.5.1	CRIBL-22716

Problem: The **Edge** > **Manage** > **Logs** UI no longer supports searching logs across entire Fleets. The option will be removed in an upcoming release.

Workaround: Search individual nodes or across Fleets in Cribl Search instead.

Unable to upgrade Leader from 4.3.0 to 4.4.0 when an external KMS is enabled

Reported in	Resolved in	Ticket
4.4.0	4.5.0	CRIBL-22299

Problem: When attempting to upgrade the Leader from 4.3.0 to 4.4.0 with an external KMS enabled, you will receive several `Secret failed to decrypt` errors and the web UI will fail to load. This is due to an issue with how `client.secret` is handled when an external KMS is enabled.

Workaround: None identified.

Notification targets created in one Cribl product can be deleted from another without warning

Reported in	Resolved in	Ticket
3.1.0	4.5.1	CRIBL-22292

Problem: When you create a Notification target, it will be made available to all products in the Cribl suite. When you attempt to delete a target from the same application you created it in (for example, Cribl Stream), the product will properly prevent the deletion when there is an active notification using the target. However, if you switch to a new application (for example, Cribl Search), you will be allowed to delete the target without error or warning. This will leave the active notification in the original product (Stream) with an undefined target. You should not be able to delete the target if it is used by any active notification in any Cribl suite product.

Workaround: None identified.

Packs installed in Fleets are not visible to Subfleets.

Reported in	Resolved in	Ticket
4.0.0	4.5.0	CRIBL-19676

Problem: When a Fleet has Subfleets, and you install a Pack in the parent Fleet, none of the Subfleet's Pipeline drop-downs make the Pack available. This is true for Routes, pre-processing Pipelines in Sources, and post-processing Pipelines in Destinations.

Workaround: Install the Pack both in the parent Fleet, and in any Subfleets that need to use the Pack.

Product-level Editor Permissions do not allow a commit/deploy action on the Worker Groups/Fleets page

Reported in	Resolved in	Ticket
4.2.0	4.5.0	CRIBL-18970, CRIBL-20113

Problem: Users with product-level Edge **Editor** or Stream **Editor** Permissions are unable to commit and deploy changes made to Worker Groups or Fleets from the Group or Fleet page. However, if the user navigates to the Group or Fleet where the change was saved, the **Commit and Deploy** button is available.

Workaround: Commit and deploy from the Worker or Group where you saved the change.

Special characters in Source or Destination auth tokens cause authentication failure

Reported in	Resolved in	Ticket
4.4.0	4.5.1	CRIBL-17047

Problem: When you include special characters in a Source or Destination auth token, those characters may cause an authentication failure.

Workaround: Use only alphanumeric characters or the underscore (`_`) in a Source or Destination auth token. Do not use any of the following characters: `<`, `>`, `"`, `` ` ``, `\r`, `\n`, `\t`, `{`, `}`, `|`, `\`, `^`, or `'`. Alternatively, upgrade to Cribl Edge 4.5.1.

Splunk Load Balanced Destination degradation with acks enabled

Reported in	Resolved in	Ticket
3.4.1	4.5.0	CRIBL-12066

Problem: On a Splunk Load Balanced Destination, enabling the **Minimize in-flight data loss** (acknowledgments) field can cause high CPU drain and backpressure.

Workaround: On the Splunk LB Destination's **Advanced Settings** tab, disable **Minimize in-flight data loss**.

Resolved in 4.4

The following issues were resolved in version 4.4 and its maintenance releases.

Windows Event Forwarder Source used up its allowed quota of connections

Reported in	Resolved in	Ticket
4.0.0	4.4.4	CRIBL-21965

Problem: When the Windows Event Forwarder (WEF) Source could not process a request coming from a Windows client, it failed to correctly close the socket or send an error response to the client. Eventually, maximum allowable active connections could be reached, stopping the flow of data.

Workaround: For any affected WEF Source, set **Advanced Settings** > **Max active requests** to `0` to allow unlimited requests. Note that this will eventually cause other problems as the number of open-inactive sockets

grows, meaning that the Worker nodes will eventually need to be restarted.

Sources incorrectly report failure

Reported in	Resolved in	Ticket
4.0.0	4.4.4	CRIBL-21690

Problem: When using OpenTelemetry HTTP, the Source always listens on the default address (either `0.0.0.0` or `::`, or both, depending on the network configuration of the host), ignoring the option you configured in **General Settings** > **Address**.

Workaround: None identified.

Certain Sources and Destinations are unavailable in Edge and Worker instances

Reported in	Resolved in	Ticket
4.4.0	4.4.2	CRIBL-21443

Problem: Some Sources and Destinations, notably Cribl HTTP and Cribl TCP, report an error of being "disabled in the current deployment" when attempting to edit their configuration from the teleport view or by direct UI access.

Workaround: Instead of modifying configurations for Edge or Worker nodes from the teleport view or using direct UI access, only use Leader nodes to modify configurations for Worker nodes.

Azure Data Explorer Destination exposes irrelevant PQ settings in Cribl.Cloud

Reported in	Resolved in	Ticket
4.4.0	4.4.4	CRIBL-21335

Problem: For Edge Nodes in Cribl.Cloud with Azure Data Explorer (ADX) Destinations: such Destinations' **Persistent Queue Settings** should expose only a **Clear Persistent Queue** button. They are incorrectly also exposing the PQ settings required for on-prem deployments.

Workaround: In Cribl.Cloud ADX Destinations, ignore all Persistent Queue settings except the **Clear Persistent Queue** button.

"Unsupported system page size" error when upgrading the Leader

Reported in	Resolved in	Ticket
4.4.0	4.4.4	CRIBL-21289

Problem: Fedora users can experience an "unsupported system page size" error when upgrading the Leader. This is due to an issue with `jemalloc`.

Workaround: You can manually remove `jemalloc.so`. Docker users can use the following:

```
docker ... -e CRIBL_BEFORE_START_CMD_0="rm /opt/cribl/bin/libjemalloc.so" ...
```

Go version update required

Reported in	Resolved in	Ticket
4.4.0	4.4.2	CRIBL-21155

Problem: The Go version requires an update.

Workaround: None identified.

Azure Data Explorer Destination fails to validate database settings in Streaming mode

Reported in	Resolved in	Ticket
4.4.0	4.4.2	CRIBL-21123

Problem: When Azure Data Explorer Destination is in **Streaming** mode and **Validate database settings** is turned on, database validation will fail unless **Ingestion service URI** has been set in **Batching** mode first.

Workaround: Switch to **Batching** mode, set the **Ingestion service URI** field, and save the Destination. Then switch back to **Streaming** mode and save the Destination again.

Stream Groups' Members indicators always show zero count

Reported in	Resolved in	Ticket
4.4.0	4.4.4	CRIBL-21099, CRIBL-22072

Problem: On a Stream Worker Group's **Manage** > **Overview** page, the **Members** counters show `0` in all Roles/Permissions, even when Members have been added.

Workaround: Navigate to **Group Settings** > **Members** for an accurate view of all Members and their access levels.

Worker Node generates an incomplete diagnostic bundle when Include git is enabled

Reported in	Resolved in	Ticket
4.4.0	4.4.2	CRIBL-21093

Problem: When you teleport to a Worker Node and create a diagnostic bundle, if you leave **Include git** toggled on in the **Create & Export Diag Bundle** modal, the bundle that the system creates will be malformed and missing information, and the UI will produce a fatal error notification.

Workaround: When you create diagnostic bundles, turn **Include git** off.

Azure Data Explorer Destination does not correctly turn off Validate database settings

Reported in	Resolved in	Ticket
4.4.0	4.4.2	CRIBL-21060

Problem: Once the Azure Data Explorer Destination has been saved with **Validate database settings** on, turning the setting off has no effect, and the Destination performs validation anyway.

Workaround: To ensure that **Validate database settings** is turned off, perform either of these two workarounds:

Option 1: Clone the Destination or create a new one, and turn **Validate database settings** off before saving.

Option 2: Turn off **Validate database settings**, then restart the Cribl Edge Edge Node.

Rapid logging causes Edge Node memory spikes and prevents logging

Reported in	Resolved in	Ticket
4.3.0	4.4.2	CRIBL-20607

Problem: A large number of logs written in a short time causes memory spikes in Edge Nodes and stops log output.

Workaround: None identified.

Upgrading to 4.3.0 or later causes Amazon Kinesis Data Streams and WEF Sources to lose state

Reported in	Resolved in	Ticket
4.3.0	4.4.2	CRIBL-20444

Problem:

After an upgrade to 4.3.0, Edge Nodes for a Cribl Edge Amazon Kinesis Data Streams Source or Windows Event Forwarder (WEF) Source can lose state upon init. This can be especially problematic for customers with large streams and high data ingestion.

An Amazon Kinesis Data Streams Source that loses state will fail to read a data stream from the point where it most recently left off. Instead, the Source will start reading from the location configured by **Optional Settings** > **Shard iterator start**.

- If **Shard iterator start** is set to `Earliest Record` (the default), the Source can start too far behind in the stream to ever catch up.
- If **Shard iterator start** is set to `Latest Record`, the Source can start later than where it left off, causing some records to be skipped.

A Microsoft Windows Event Forwarder (WEF) Source (with bookmarks configured for the subscription) that loses state will fail to request that upstream clients send events starting at a bookmarked location. Instead,

Cribl Edge will send no bookmark, prompting upstream clients to send **all** events that otherwise match the subscription. This can cause Cribl Edge to ingest duplicate events from WEF clients.

Workaround: If you have not yet upgraded to 4.3.0 or later, consider delaying the upgrade until this issue is fixed.

After upgrading, Monitoring dashboards are not visible for users with `GroupRead` and `GroupCollect` Policies

Reported in	Resolved in	Ticket
4.2.0	4.4.3	CRIBL-18925

Problem: Users that were assigned `GroupRead` and `GroupCollect` Policies in Cribl Edge 4.1.0 will no longer be able to access the **Monitoring** tab and dashboard after upgrading to Cribl Edge 4.2.

Workaround: For `GroupRead`, migrate the user to **Read Only** Permission at the Group-level in Members and Permissions. Assign the Member as Stream **User** then set their Permissions for the group as **Read Only**. For `GroupCollect`, there is no specific workaround but users can be assigned the `GroupEdit` Policy if they need to view Monitoring dashboards.

Unable to bind `CRIBL_DIST_MASTER_URL` to an IPv6 address

Reported in	Resolved in	Ticket
4.1.0	4.4.0	CRIBL-16284

Problem: Due to an issue with the `CRIBL_DIST_MASTER_URL` environment variable, Cribl will not bind to a specified IPv6 address.

Workaround: Manually update the settings in the two relevant configuration files, `instance.yml` and `cribl.yml`, for RPC and UI communication respectively.

For the UI, manually define the host setting in the `cribl.yml` file:

```
api:
  host: "::"
```

To configure the instance as a Leader node and listen on all IPv6 and IPv4 addresses, manually configure the `instance.yml` file:

```
distributed:
  mode: master
  master:
    host: "::"
    port: 4200
    tls:
      disabled: true
    authToken: criblmaster
```

Cascading problems when Windows Event Logs Source collects from `ForwardedEvents` channel

Reported in	Resolved in	Ticket
3.5.0	4.4.0	CRIBL-14869

Problem: Using the Windows Event Logs Source to collect events from a Windows `ForwardedEvents` channel can cause misattribution of logs to the local machine that hosts Cribl Edge, and undercollection of local logs from other logging channels (such as Security, System, or Application).

Workaround: Exclude the `ForwardedLogs` channel from your **Event Logs** selection.

Resolved in 4.3

The following issues were resolved in version 4.3 and its maintenance releases.

Microsoft Windows Installer (MSI) does not create an Edge desktop shortcut

Reported in	Resolved in	Ticket
4.3.0	4.3.1	CRIBL-20504

Problem: When you install Cribl Edge on Windows, the MSI doesn't create the expected desktop shortcut for Edge, even if you select **Create a shortcut for Cribl Edge on the desktop**.

Workaround: None identified.

Scheduled (cron-based) jobs fail to load for some Sources and Collectors

Reported in	Resolved in	Ticket
4.3.0	4.3.1	CRIBL-20088

Problem: Collection tasks for Office 365 Sources, the Splunk Search Source, the Prometheus Scraper Source, and scheduled Collectors fail to load when the Leader restarts, fails over, or upgrades. Collectors running jobs ad hoc are not affected.

Workaround: If you're using one of the listed Sources or use scheduled Collector jobs on-prem, do not upgrade to 4.3.0. If you are already on 4.3.0, downgrade to 4.2.2, or wait and upgrade to 4.3.1 when it's available.

You can also restart the config helper for on-prem installations using the command: `pkill -f CONFIG_HELPER`

You must run this command each time the Leader fails over, or is restarted (ideally, not too often).

Chain Function's link breaks when the chained Pipeline/Pack is resaved

Reported in	Resolved in	Ticket
4.3.0	4.3.1	CRIBL-20083

Problem: After linking a Chain Function to a Pipeline or Pack, resaving the target Pipeline/Pack triggers `Invalid link` errors.

Workaround: 1. Copy the Pipeline containing the Chain Function to your clipboard, delete the original Pipeline, and then paste the copy to resolve the error. 2. Alternatively, on-prem admins can restart the config helper that manages the Fleet. (In the UI, navigate to **Settings** > **Global** > **System** > **Services** > **Processes**.)

In Cribl.Cloud, Admin Members cannot modify Global Settings

Reported in	Resolved in	Ticket
4.3.0	4.3.1	CRIBL-20038

Problem: After upgrading to 4.3.0, Members granted the Admin Permission cannot modify Global Settings.

Workaround: Members with the Owner Permission can still modify Global Settings.

Edge Nodes cannot be upgraded in Leader-managed upgrades

Reported in	Resolved in	Ticket
4.3.0	4.3.1	CRIBL-20086

Problem: In Stream and Edge, if you upgrade the Leader to 4.3.0, then use **Commit & Deploy** to push configs to Fleets before upgrading Edge Nodes to 4.3.0, you won't be able to upgrade the Edge Nodes from the Leader thereafter.

This issue affects only Leader-managed upgrades of Edge Nodes.

Workaround: If a Leader is on 4.3.0, you should upgrade all Edge Nodes to 4.3.0 before making any config changes or committing and deploying to the Nodes.

If you have already committed and deployed to Edge Nodes that are on a version prior to 4.3.0 (and the Leader is on 4.3.0), here are three workaround options:

- Revert and redeploy the last commit, then upgrade the Edge Nodes and deploy the most up-to-date changes, **OR**
- Upgrade the Edge Nodes via command line or script, **OR**
- Downgrade the Leader to 4.2.2 and then upgrade to 4.3.1 when it is available.

If you run into issues, contact support@cribl.io for resolution assistance.

Any Code Function before another Function breaks the Data Preview OUT tab display

Reported in	Resolved in	Ticket
4.3.0	4.3.1	CRIBL-20040

Problem: Any Code Function inserted before the end of a Pipeline prevents the Data Preview OUT tab from rendering properly. This occurs even if the Code Function contains only a single comment, or is blank. Adding a Code Function at the end of a Pipeline does not cause an issue. This is a Data Preview UI issue only – if you capture events to a Destination, Cribl Edge processes them as expected.

Workaround: None identified.

Numeric sub-second values sent over S2S v4 require care to avoid incorrect stringification

Reported in	Resolved in	Ticket
4.0.0	4.3.1	CRIBL-20017

Problem: Events received from S2S v4 can have sub-second granularity, meaning that the Splunk TCP Source reads a sub-second value from ingested events and uses it to set a `_subsecond` field added to outgoing events. If, for example, the Source sets the `_subsecond` field with a stringified numeric value instead of the actual number, problems can occur that affect downstream receivers such as Splunk indexers.

Workaround: Either of the following workarounds can be effective:

1. Upgrade to Cribl Edge 4.3.1 or newer.
2. In a Pipeline, drop or numerify any `_subsecond` field whose value is a string.

When in XML format, the Windows Event Logs Source doesn't accept log names that contain spaces

Reported in	Resolved in	Ticket
4.2.2	4.3.0	CRIBL-19818

Problem: When the Windows Event Logs Source is set to XML format, any configured log names that contain spaces (e.g., `Windows Powershell`) cause an error and the log can't be read.

Workaround: None identified.

PQ doesn't drain for the Event Hubs Destination when Acknowledgements is set to All or Leader

Reported in	Resolved in	Ticket
4.2.2	4.3.1	CRIBL-19756

Problem: When the `Acknowledgements` setting is set to `All` or `Leader`, Kafka-based Destinations (especially Azure Event Hubs) can fail to drain the persistent queue (PQ) and the logs are filled with `Attempting to send faster than the downstream can receive – consider checking the network connection and broker health` errors.

Workaround: You can set a limit using the `Persistent Queue Drain rate limit (EPS)` setting (for example, set it to `500`) to get the PQ to drain when the datagen is turned off. However, if the rate of data input continues to exceed the rate at which the Destination can send events downstream (which is slowed by the `Acknowledgements` setting), the PQ will continue to build. Alternatively, you can set the `Acknowledgements` setting to `None`, which significantly increases the rate at which data can be sent to Event Hubs.

Deleting a Source via QuickConnect doesn't refresh the view

Reported in	Resolved in	Ticket
4.2.0	4.3.0	CRIBL-19709

Problem: When using QuickConnect, if you delete a Source by opening its drawer and clicking `Delete Source`, the Source is deleted. However, it will continue to show on the page until you manually refresh the page.

Workaround: Manually refresh the page after deleting a Source.

When adding or editing a Route, typeahead misbehaves in the Filter field

Reported in	Resolved in	Ticket
4.2.2	4.3.0	CRIBL-19571

Problem: Entering text in the **Filter** field of a Route is difficult in Cribl Edge 4.2.2 because the typeahead function behaves erratically, sometimes moving the cursor around or inserting text fragments.

Workaround: 1. Assemble the filter expression in a different Cribl expression field, or in any text editor, then copy and paste it here. 2. Alternatively, click the Manage as JSON button at the **Data Routes** table's upper right.

Then, in the JSON editor, build the expression as the `"filter":` key's value.

Group Information for Cribl.Cloud Worker Groups not showing on summary page for SSO users

Reported in	Resolved in	Ticket
4.2.2	4.3.0	CRIBL-19107

Problem: On Cribl.Cloud Organizations, when you open a Stream Group's **Manage** > **Overview** page, the **Group Information** section is not populated for users imported using SSO from external identity providers. (This section populates correctly for users/Members configured natively within your Cribl Organization.)

Workaround: None identified.

When configuring an S3 Collector, JavaScript expressions break the "Auto-populate from" option for the Path field

Reported in	Resolved in	Ticket
2.3.0	4.3.0	CRIBL-18844

Problem: When automatically populating an S3 Collector from an S3 Destination, the Collector **Path** field won't resolve JavaScript expressions, even if the expression was valid on the Destination that the field was populated from. The S3 Collector path is treated like a literal string and the software fails to warn you that the path is invalid.

Workaround: None identified.

When configuring an HTTP Source, enabling Source PQ changes the inputId in events

Reported in	Resolved in	Ticket
4.1.3	4.3.0	CRIBL-18668

Problem: When Source-side Persistent Queueing is enabled on an HTTP Source, the trailing colon is dropped from `__inputId`. This will break Routes and filters that are based on the Input ID if it is copied from the configuration page. Live captures will also stop working until you modify them.

Workaround: None identified.

Default Pack is not visible to Project Editor when a Project is shared with them

Reported in	Resolved in	Ticket
4.2.0	4.3.0	CRIBL-18639

Problem: When a Cribl admin sets up a Project, a default Pack is included. If the Admin then shares that Project with a Project editor, the editor will not see the default Pack.

Workaround: In the Project view, the Project editor can add any Pack that has been made available by the admin, including the default Pack.

Users with Group-level Read Only Permissions can interact with logging level menus

Reported in	Resolved in	Ticket
4.2.0	4.3.1	CRIBL-18556

Problem: Users with the **Read Only** Permission at the Group-level can access the **Add Channel** and **Delete** buttons under **Group** > **Manage** > **Group Settings** > **Logging** > **Levels**. However, they are unable to save changes. When performing these actions, **Read Only** users will see a **Forbidden** message displayed in the UI and no changes will be saved. The setting should not be accessible to users with this Permission.

Workaround: None identified.

HTTP Destinations sometimes send oversized payloads

Reported in	Resolved in	Ticket
4.2.0	4.3.0	CRIBL-18218

Problem: HTTP-based Destinations do not strictly enforce the **Max Body Size** limit in all cases. This can cause payloads to be sent that exceed the maximum size some downstream receivers can accept.

Workaround: Experiment with lower **Max Body Size** values until downstream receivers are reliably accepting all events.

If a deployment fails, Workers will not automatically revert to the previous version

Reported in	Resolved in	Ticket
4.1.0	4.3.1	CRIBL-16203

Problem: When a deployment fails, Workers cannot revert `default/cribl` to a previous version because that directory is no longer backed up. The Worker will enter a broken state if you provide an invalid deploy bundle, because it cannot revert to the last valid state.

Workaround: To resolve any broken Workers, you must deploy a valid configuration. The Worker will resume working once it picks up the new configuration.

Cribl Edge/Windows Limitation: Cannot upgrade Edge Nodes from the Leader's UI

Reported in	Resolved in	Ticket
3.5.0	4.3.0	CRIBL-10870

Problem: Cribl Edge/Windows does not support upgrading Edge Nodes via the Leader's UI.

Workaround: Rerun the Windows installer on each Edge Node, specifying the same installation options/parameters you used when installing the preceding version.

Resolved in 4.2

The following issues were resolved in version 4.2 and its maintenance releases.

Inherited Packs in Fleets lose their configured Routes

Reported in	Resolved in	Ticket
4.0.0	4.2.0	CRIBL-19700

Problem: Routes in a Pack at the Fleet level revert to default Routes when inherited by a SubFleet.

Workaround: None identified.

File Monitor Source causes memory leaks and Edge Node crashes

Reported in	Resolved in	Ticket
4.2.0	4.2.2	CRIBL-19154

Problem: The File Monitor Source, when running in Manual Discovery mode, leaks memory and a file descriptor each time it rediscovers a file that it has already collected. The leaked resources increase with every polling interval, for every rediscovered file, eventually causing Edge Nodes to crash. This does not affect Auto Discovery mode, and does not affect deployments without an active File Monitor Source.

Workaround: File Monitor users should bypass (or roll back from) versions 4.2.0–4.2.1.

AppScope Source might reference a nonexistent config

Reported in	Resolved in	Ticket
4.2.0	4.2.2	CRIBL-19044

Problem: The 'sample_config' stock configuration was removed in 4.2.0. This missing config causes the Appscope Source to disappear from the UI, and changes to other Sources also fail with validation errors.

Workaround: For on-prem deployments, clone one of the existing AppScope configurations and rename it to `sample_config`. For Cribl.Cloud customers, there is no workaround yet.

Users with the Admin Permission cannot view Notifications

Reported in	Resolved in	Ticket
4.2.0	4.2.2	CRIBL-19015

Problem: Users with the **Admin** Permission receive a `You do not have sufficient permissions to access this resource` message when attempting to view Notifications. Users with this Permission should be able to access Notifications.

Workaround: None identified. Organization **Owners** can access these notifications.

Users with the Admin Permission cannot bootstrap a Worker due to a missing auth token

Reported in	Resolved in	Ticket
4.2.0	4.2.2	CRIBL-19014

Problem: Users with the **Admin** Permission can't add a Worker using **Add / Update Worker Node**, because the modal doesn't have the **Leader hostname/IP** or the **Auth token** fields populated. You can enter the hostname/IP into its field, but not the auth token. Users with the **Admin** Permission should see the auth token.

Workaround: Organization **Owners** can perform this action and share the auth token.

Credential encryption errors preventing user access to Cribl Edge Sources and Destinations

Reported in	Resolved in	Ticket
4.2.0	4.2.1	SAAS-4823

Problem: A critical regression caused Fleets to become unresponsive, preventing access to Sources and Destinations. This issue was caused by incorrectly encrypted credentials.

Workaround: Upgrade to Cribl Edge 4.2.1.

Users with the Edge Editor Permission cannot delete secrets under Fleet Settings

Reported in	Resolved in	Ticket
4.2.0	4.2.2	CRIBL-18969

Problem: Users with the Edge **Editor** Permission can create secrets under **Fleet** > **Fleet Settings** > **Secrets** but they are unable to delete the secret once it is created. Edge **Editors** should have this capability.

Workaround: Assign the Edge **Admin** Permission to users that need to be able to delete secrets.

Edge Node Settings not visible when teleporting

Reported in	Resolved in	Ticket
4.2.0	4.2.2	CRIBL-18963

Problem: When you teleport into an Edge Node, you can't view **Node Settings** in the UI.

Workaround: None identified.

Users with the Edge Editor Permission cannot delete Sources, Destinations, or Pipelines

Reported in	Resolved in	Ticket
4.2.0	4.2.2	CRIBL-18951

Problem: If you assign a user the Edge **Editor** permission at the product-level, the user will not be able to delete Sources, Destinations, or Pipelines. Edge **Editors** should have this capability.

Workaround: Assign the Edge **Admin** Permission to users that need to be able to delete these items.

Change from Edge User to Edge Editor can fail to take effect

Reported in	Resolved in	Ticket
4.2.0	4.2.2	CRIBL-18950

Problem: If you assign the **Editor** Permission to an Edge **User** at the Fleet-level and then later assign the Edge **Editor** Permission to the same User at the product-level (Edge **User** now becomes an Edge **Editor**), the **Editor** Permission at the product-level can fail to take effect. The user will have **Editor** access to the one Fleet at the Fleet-level but they will still show as an Edge User and will not have **Editor** Permissions at the product-level.

Workaround: None identified.

Product logins for users with product-level Read Only and Editor Permissions can result in misleading log entries

Reported in	Resolved in	Ticket
4.2.0	4.2.2	CRIBL-18948

Problem: When users that are assigned product-level **Read Only** or **Editor** Permissions log in to a product, the browser incorrectly makes requests for resources that users at this level are not allowed to access. While these requests are correctly denied and there is no risk of compromising security, Cribl admins may see misleading log entries indicating that these users tried to access something they are not allowed to access.

Workaround: None identified.

Edge users with the Fleet-level Editor Permissions cannot use manual file discovery

Reported in	Resolved in	Ticket
4.2.0	4.2.2	CRIBL-18923

Problem: When using manual file discovery in Cribl Edge, users that have been assigned the **Editor** Permission at the Group/Fleet-level will not see any results.

Workaround: You can assign the **Admin** Permission for users that require the manual file discovery feature.

Stream Users are able to access the Monitoring page

Reported in	Resolved in	Ticket
4.2.0	4.2.2	CRIBL-18848

Problem: Stream **Users** with no Group-level Permissions are able to access the **Monitoring** page. The **Monitoring** menu item is hidden, but the page can be access manually using a URL.

Workaround: None identified.

Recent Actions endpoint returns results for certain unauthorized users

Reported in	Resolved in	Ticket
4.2.0	4.2.2	CRIBL-18818

Problem: The Recent Actions endpoint `GET /api/v1/ui/recentActions` returns more recent actions than intended for non-admin users.

No Access to Subfleets for the Edge User Permission

Reported in	Resolved in	Ticket
4.2.0	4.2.2	CRIBL-18784

Problem: Regardless of what Permissions are granted at the Fleet-level, Edge members assigned to the **User** Permission will not have access to Subfleets.

Workaround: At the (Edge) product level, you must grant your members **Read Only**, **Editor**, or **Admin** Permissions so that they can access Subfleets.

(Linux) System Metrics Source does not emit “Per interface” metrics when set to “All”

Reported in	Resolved in	Ticket
3.5.0	4.2.0	CRIBL-18473

Problem: In the (Linux) **System Metrics** Source > **Host Metrics** tab, selecting **All** does not emit **Per Interface** metrics.

Workaround: In the **Host Metrics** tab, select **Custom** > **Network** > **Custom** then toggle **Per interface metrics** to `Yes`.

Project editor can’t delete Pipelines via Options menu

Reported in	Resolved in	Ticket
4.2.0	4.2.2	CRIBL-18447

Problem: In a Project’s **Pipelines** modal, when a Project editor opens a Pipeline’s ••• (Options) menu, the **Delete** option is unavailable. Cribl Edge displays a spurious error about insufficient permissions.

Workaround: Use the list’s check boxes to select Pipelines, then click **Delete Selected Pipelines**.

High CPU usage in File Monitor Source’s Manual mode

Reported in	Resolved in	Ticket
4.1.3	4.2.0	CRIBL-18400

Problem: In the **File Monitor** Source and the **Explore** > **Files** tab UI, the **Manual** mode does not honor the **Max depth** setting. The API process consumes high CPU resources because the discovery logic (used in the **File Monitor** Source and **Files** tab) recurses in the directory tree.

Workaround: None identified.

Using the Rename Function gives unexpected results due to skipped internal fields

Reported in	Resolved in	Ticket
3.5.2	4.2.0	CRIBL-18285

Problem: When internal fields are present in the `Rename fields` list, the Rename Function may incorrectly assign field keys or values.

Workaround: This function is not intended to operate on internal fields. Avoid this operation.

Amazon CloudWatch Destinations log error while flushing events older than 24 hours

Reported in	Resolved in	Ticket
4.1.3	4.2.0	CRIBL-18184

Problem: Amazon CloudWatch Destination doesn't filter batches of events to ensure all events in the batch are within 24 hours of each other as required by the API, resulting in the batch being rejected by Cloudwatch.

Workaround: Add a Post-Processing Pipeline with a Drop Function using a Filter Expression of `_time<(Date.now() / 1000) - 86400`.

The Office 365 Activity Source misses events

Reported in	Resolved in	Ticket
4.1.2	4.2.0	CRIBL-18164

Problem: The Office 365 Activity Source misses events due to the current collection system. For services such as Exchange, SharePoint, OneDrive, and Teams, Microsoft indicates that audit record availability is typically 60 to 90 minutes after an event occurs. Our current collector methodology does not account for this availability window. Instead, the Source completes runs as scheduled and collects only events it finds during the configured time range, which can lead to missed events.

Workaround: None identified.

GroupEdit is not able to commit changes

Reported in	Resolved in	Ticket
4.0.0	4.2.0	CRIBL-17939

Problem: Users having the `GroupEdit` policy are able to make changes (e.g. create new Sources) but are unable to **Commit** their changes. These users should be able to **Commit**, but not **Deploy**, changes.

Workaround: Apply the `GroupFull` policy for users that need to be able to **Commit** changes. These users will also have the ability to deploy a Worker Group or Fleet.

Edge Nodes don't update from Cribl Edge Leader

Reported in	Resolved in	Ticket
4.1.2	4.2.0	CRIBL-17792

Problem: In Cribl Edge 4.1.2–4.1.3, if you define the `baseURL` key on a Leader, Edge Nodes don't get the latest config version from the Leader. The log may also contain `checksum mismatch` warnings.

Workaround: Clear **URL base path** in **Settings** > **General Settings** > **Advanced** or specify an empty `baseURL` key in `cribl.yml`. If you need to define a base URL, do not upgrade to the affected versions.

Windows Event Forwarder Source causes duplicate events when experiencing backpressure

Reported in	Resolved in	Ticket
4.0.0	4.2.0	CRIBL-17614

Problem: In certain cases where the Windows Event Forwarder Source experiences backpressure from a downstream Pipeline or Destination, it will return an incorrect error message to the client that sent the events.

This causes that client to re-send the same events and results in duplicate events ingested in Stream.

Workaround: In cases where the cause of the backpressure can be resolved, this issue is mitigated. However if the backpressure cannot be completely removed, there is no additional workaround.

Kafka-based Destinations fail to report backpressure properly

Reported in	Resolved in	Ticket
4.1.0	4.2.2	CRIBL-16944

Problem: With a downstream receiver exerting backpressure, Kafka-based Destinations do not report backpressure promptly, even though they effectively exert backpressure. As a result, upstream data might stall for several minutes before appropriate errors surface in the Destination's health status, and PQ (if configured) will not engage promptly.

Workaround: Decreasing the value for the `Request timeout (ms)` setting within the `Advanced Settings` section of the destination configuration might lead to the Destination reporting backpressure more promptly.

The Chain Function has a 10% impact on performance

Reported in	Resolved in	Ticket
4.1.0	4.2.2	CRIBL-15538

Problem: Using the Chain Function to chain data processing from one Pipeline or Pack to another degrades performance by about 10%, compared to running the original Pipeline or Pack directly.

Workaround: None identified.

Azure Event Hubs Destination with PQ drops events when inbound data is interrupted

Reported in	Resolved in	Ticket
3.5.3	4.2.0	CRIBL-17661, replaces CRIBL-12649

Problem: When Azure Event Hubs (and other Kafka-based Destinations) have persistent queues (PQs) configured, an interruption of inbound data flow (e.g., due to network issues) can cause the Destination to start dropping events.

Workaround: On the Source whose events are being interrupted, configure an Always On persistent queue. This buffering will cause the Destination to drop fewer events. Note that persistent queuing will not engage until Cribl Edge has exhausted all attempts to send the data to the Kafka receiver.

Resolved in 4.1

The following issues were resolved in version 4.1 and its maintenance releases.

AES-256-GCM security vulnerability

Reported in	Resolved in	Ticket
4.1.2	4.1.3	CRIBL-18038

Problem: The AES-256-GCM encryption option introduced in 4.1.2 included a security vulnerability.

Workaround: Do not use this option in 4.1.2.

Kubernetes Logs Event Breaker Incorrectly Breaks Events

Reported in	Resolved in	Ticket
4.1.0	4.1.3	CRIBL-17907

Problem: The **Kubernetes Logs** Event Breaker incorrectly breaks events with out-of-order timestamps.

Workaround: In the Event Breaker, move the logic that adjusts `_raw` into a **Pre-Processing** Pipeline.

Perf bug: RPC consuming excessive CPU compiling expressions

Reported in	Resolved in	Ticket
4.1.1	4.1.3	CRIBL-17779

Problem: In 4.1.1, to improve product security, we changed how we manage our RPC traffic from Leaders and Workers. This change can have a negative impact in larger environments with many Worker Processes. Following a 4.1.1 or 4.1.2 upgrade, the Leader can become non-responsive to UI requests due to a steady 100% CPU utilization. When this happens, you cannot manage or monitor Cribl Edge environments.

Workaround: Roll back to a previous stable version, such as 4.1.0.

CrowdStrike Destination's "LogScale endpoint" field hidden from UI

Reported in	Resolved in	Ticket
4.1.2	4.1.3	CRIBL-17715

Problem: The CrowdStrike Falcon LogScale Destination's **LogScale endpoint** field is hidden from the 4.1.2 configuration modal, but can be restored.

Workaround: Follow these steps, in sequence, for each LogScale Destination. 1. Populate the LogScale config modal, and save once. 2. Reopen the modal, then select **Manage as JSON**. 3. Change the `"loadbalanced":` key's value from `true` to `false`. 4. Copy this default nested element: `"url": "https://cloud.us.humio.com/api/v1/ingest/hec",` 5. Paste it above the whole `"urls"` element (typically at line 21). 6. Click **OK** to restore the visual UI, with the **LogScale endpoint** field now visible. 7. Enter your actual endpoint URL, and save the config.

Expanding Status of Output Router Destination triggers error

Reported in	Resolved in	Ticket
4.1.1	4.1.3	CRIBL-17632

Problem: When you attempt to expand a node on the **Status** tab for an Output Router Destination, the page crashes and the error `Cannot convert undefined or null to object` appears.

Workaround: None identified.

The Kubernetes Logs Source drops events

Reported in	Resolved in	Ticket
4.0.0	4.1.3	CRIBL-17602

Problem: Events from a running container will stall when the underlying container runtime rotates the log file. Any further rotation of log files will result in data loss.

Workaround: Restart the Kubernetes Logs Source to reconnect.

S3 ingestion slows after upgrade

Reported in	Resolved in	Ticket
4.1.1	4.1.2	CRIBL-17317

Problem: After upgrading to Cribl Edge 4.1.1, ingesting using any `JSON` Array Event Breaker (such as AWS CloudTrail) will slow or stop due to high CPU usage.

Workaround: Roll back to a previous stable version, such as 4.1.0.

Edge Nodes integrated with `systemd/initd` can fail upon reconfig

Reported in	Resolved in	Ticket
4.1.1	4.1.2	CRIBL-17264

Problem: Where 4.1.1 customer-managed (hybrid or on-prem) Edge Nodes (on Linux) are integrated with `systemd/initd`, deploying new config bundles to those Edge Nodes can stop the Edge Nodes from processing data. The root cause is an incorrect backup step, which (depending on permissions) can also consume large amounts of disk space on their hosts.

Precondition: Cribl-managed Cloud instances are unaffected. This problem has been observed only if at least one of the following directories is populated, or exists with restricted permissions:

```
/data/
/default/cribl/
/default/edge/
/local/cribl/
/local/edge/
/default/<any-installed-Pack-name>/
/local/<any-installed-Pack-name>/
```

Workaround: Skip 4.1.1 or roll back to your last stable version. To run 4.1.1, this workaround is available on systemd's `Service` section: 1. In the script `/etc/systemd/system/cribl.service`, add the line: `WorkingDirectory=/opt/cribl`, (This is the default path – specify your own path equivalent to

`$CRIBL_HOME`). 2. Then reload the systemd daemon: `systemctl daemon-reload`. 3. Then restart the Cribl Stream instance using `systemctl restart <service name>`.

Requests from Office 365 Message Trace Source failed intermittently

Reported in	Resolved in	Ticket
2.2.0	4.1.0	CRIBL-16929

Problem: Microsoft has fixed this Office 365 Message Trace API problem with this patch (Microsoft login required). Before this patch, requests failed intermittently, often with `Non-whitespace` errors.

Workaround: None identified.

Monitoring incorrectly shows metrics for disconnected Subscriptions

Reported in	Resolved in	Ticket
4.1.1	4.1.2	CRIBL-16926

Problem: Selecting **Monitoring** > **Data** > **Subscriptions** will falsely show statistics even for Subscriptions that are not connected to a Destination, and therefore have no data flow.

Workaround: Ignore these Monitoring statistics.

Windows Event Logs Source should have a configurable Max Event Size

Reported in	Resolved in	Ticket
4.1.0	4.1.1	CRIBL-16549

Problem: Events aren't broken properly when collecting logs from the PowerShell event log. The Source contains a Max Event Limit of `51200`, which is a hard-coded breaker config. This limit should be configurable.

Workaround: None identified.

Cribl.Cloud API Credential's Roles aren't honored on tokens

Reported in	Resolved in	Ticket
4.1.0	4.1.2	SAAS-3681

Problem: API Bearer tokens obtained through the Cribl.Cloud portal are all granted the `Admin` Role, regardless of the scope specified on the parent Credential.

Workaround: Use the pre-4.0 workaround to obtain Bearer tokens from the in-app API Reference.

Splunk Load Balanced Destination incorrectly re-creates all connections during DNS resolution

Reported in	Resolved in	Ticket
4.1.0	4.1.1	CRIBL-16494

Problem: When **Minimize in-flight data loss** is enabled, the Splunk Load Balanced Destination re-creates all outbound connections during DNS resolution. This will cause the Destination to report a blocked status until the outbound connections refresh. If persistent queues (PQ) are enabled, PQ will engage while the Destination is blocked.

Workaround: Select the Destination's **Manage as JSON** option, to add the key-value pair: `"maxFailedHealthChecks": 1`.

Add Kubernetes and Docker modals in the UI aren't checking the `CRIBL_BOOTSTRAP_HOST` environment variable

Reported in	Resolved in	Ticket
4.1.0	4.1.1	CRIBL-16432

Problem: The Add Windows and Linux modals check for a bootstrap hostname to see if a user has configured a host override with the `CRIBL_BOOTSTRAP_HOST` environment variable. The Add Kubernetes and Docker modals aren't checking the environment variable as expected.

Workaround: None identified.

Leader's config deployments to pre-4.1 Workers silently fail

Reported in	Resolved in	Ticket
4.1.0	4.1.1	CRIBL-16339

Problem: A 4.1.0 Leader requires Workers to also be upgraded to – 4.1.0. If you attempt to deploy configs to Workers running earlier versions, the intended upgrade prompt might not appear. In this case, the deploy will silently hang.

Workaround: If you haven't enabled automatic upgrades of customer-managed (hybrid or on-prem) Workers, upgrade all Fleets to 4.1.0 before you deploy to them.

Cribl Stream Database Collector cannot connect with SQL Server using AD authentication

Reported in	Resolved in	Ticket
4.1.0	4.1.2	CRIBL-16304

Problem: On the indicated versions, the Database Collector cannot connect with SQL Server using Active Directory (AD) authentication. (Only local auth works.)

Workaround: Reverting to Cribl Stream 4.0.4 restores the Database Collector's ability to connect using AD authentication.

System Activity drawer on Windows Edge Nodes displays no data

Reported in	Resolved in	Ticket
3.5.0	4.1.1	CRIBL-16194

Problem: When you navigate to a Fleet's **List View** and click a row containing a Windows Edge Node, the System Activity drawer displays no data.

Workaround: None identified.

QuickConnect Pipeline Settings button fails

Reported in	Resolved in	Ticket
4.1.0	4.1.1	CRIBL-16142

Problem: When adding a new Pipeline using QuickConnect, clicking the gear ( ) button fails to open Pipeline Settings.

Workaround: After saving the Pipeline in QuickConnect, use **Manage** > **Processing** > **Pipelines** to select your new Pipeline. Click the gear button here to access Pipeline Settings as expected.

Duplicate data with Event Hubs

Reported in	Resolved in	Ticket
4.0.0	4.1.1	CRIBL-16102

Problem: When **Minimize Duplicates** is enabled, the Azure Event Hubs Source assigns partitions to multiple Edge Nodes in the same Fleet, which generates duplicate data.

Workaround: Toggle **Minimize Duplicates** to `No` in **Azure** > **Event Hubs** > **Advanced Settings**.

Kafka-based Sources are omitting the `_time` field

Reported in	Resolved in	Ticket
4.0.4	4.1.0	CRIBL-15696

Problem: Kafka-based Sources (Kafka, Azure Event Hubs, Confluent Cloud) are not sending the `_time` field.

Workaround: If Cribl Edge is not retrieving the `_time` field, then in the affected Sources' config modals, use the **Processing Settings** > **Fields** tab to re-create `_time`. This new `_time` field will add the current (ingestion) time instead of the message time.

DNS Resolution reconnects Cribl TCP connections every time

Reported in	Resolved in	Ticket
4.0.0	4.1.0	CRIBL-15363

Problem: All Cribl TCP connections will reconnect every time DNS Resolution occurs, even when reconnection is not necessary.

Workaround: None identified.

Enable Automatic Upgrades deletes remote repo's Git Settings

Reported in	Resolved in	Ticket
4.0.4	4.1.3	CRIBL-15502

Problem: Enabling the Leader's **Upgrade** > **Enable Automatic Upgrade** setting deletes the corresponding remote repo's stored **Git Settings**.

Workaround: Reconfigure your repo at **Git Settings** > **Remote**.

Sending data to an inactive Kafka, Confluent Cloud, or Azure Event Hubs Destination triggers an endless series of log errors

Reported in	Resolved in	Ticket
4.0.4	4.1.1	CRIBL-15455

Problem: If you accidentally make a Kafka, Confluent Cloud, or Azure Event Hubs Destination inactive by putting a mismatched value in the **Advanced Settings** > **Environment** setting, Cribl Edge will send an endless series of errors to the log file.

Workaround: Correct the **Environment** value or leave it empty.

Logging-level changes do not take effect for services

Reported in	Resolved in	Ticket
4.0.0	4.1.1	CRIBL-15365

Problem: Changing logging levels in the Leader's Settings has no effect on the logs for the Connections, Lease Renewal, Metrics, or Notifications services.

Workaround: None identified.

On-prem Edge Nodes automatically upgrade, ignoring Leader settings

Reported in	Resolved in	Ticket
4.0.3	4.1.1	CRIBL-15367

Problem: Upgrading a Leader Node causes its customer-managed (on-prem) Edge Nodes to automatically upgrade, even when the Leader's **Enable automatic upgrades** option is set to `No` (the default). This problem does not affect Cribl-managed Edge Nodes in Cribl.Cloud.

Workaround: 1. Toggle **Enable automatic upgrades** to `Yes` and save the configuration; then slide it back to `No` and save again. Or: 2. Edit `local/cribl/cribl.yml` to explicitly set `upgradeSettings.disableAutomaticUpgrade` to `true`.

Pipeline is deselected in QuickConnect after modification

Reported in	Resolved in	Ticket
4.0.0	4.1.0	CRIBL-15246

Problem: If you add a Pipeline in the QuickConnect UI, then modify it in the **Edit Pipeline** modal, the Pipeline is deselected in the **Add Pipeline to Connection** modal. If you then close the **Add Pipeline to Connection** modal, the Pipeline will disappear from the Route.

Workaround: Select the modified Pipeline in the **Add Pipeline to Connection** modal before closing it.

Event Breaker timestamp extraction fails after configured time

Reported in	Resolved in	Ticket
4.0.0	4.1.1	CRIBL-15107

Problem: Event Breakers' timestamp extraction stops working after the Rule's configured **Future timestamp allowed** value (if set). All subsequent events received get the current time as their timestamp.

Workaround: Set a sufficiently large value for **Future timestamp allowed** – e.g., one year from the date you re/configure the rule. Alternately, restore normal timestamp extraction by restarting Worker Processes.

`CRIBL_DIST_WORKER_PROXY` env var is ignored when Leader Node's Master URL is set via `instance.yml`

Reported in	Resolved in	Ticket
3.0.3	4.1.0	CRIBL-15540

Problem: Setting the Cribl Edge Leader's Master URL via `instance.yml` causes Edge Nodes to ignore the `CRIBL_DIST_WORKER_PROXY` env var. Instead of trying to connect to the proxy configured in `CRIBL_DIST_WORKER_PROXY`, Edge Nodes will try to connect to other entities (e.g., Leader Nodes) directly, producing unexpected results.

Workaround: Set the Cribl Edge Leader's Master URL via the `CRIBL_DIST_MASTER_URL` env var, rather than via `instance.yml`.

Metrics blocklist can't be changed on a global level

Reported in	Resolved in	Ticket
4.0.0	4.1.0	CRIBL-15081

Problem: The Metrics blocklist cannot be modified from **Settings** > **Global Settings** > **General Settings** > **Limits**.

Workaround: Edit the `limits.yml` file's `metricsFieldsBlacklist` element. Add the event fields for which you want to disable metrics collection; remove any event fields for which you want to restore metrics collection.

GitOps Push/read-only confirmation banner has dead link

Reported in	Resolved in	Ticket
4.0.0	4.1.0	CRIBL-14681

Problem: After you enable GitOps `Push` mode, the resulting red confirmation banner contains a dead link labeled **GitOps Workflow**.

Workaround: To switch off `Push` mode, navigate to **Settings** > **Global** > **Git Settings**, and then set **GitOps workflow** back to `None`.

Kafka-based Sources' rebalancing is logged with exaggerated severity

Reported in	Resolved in	Ticket
4.0.2	4.1.0	CRIBL-14609

Problem: The Kafka, Azure Event Hubs, and Confluent Cloud Sources log `REBALANCE_IN_PROGRESS` events at the `error` level, even though only **frequent** rebalancing indicates a system-level or processing issue.

Workaround: Treat infrequent rebalancing events as `warn`.

Fix: Depends on a change to the underlying kafkajs library.

Edge Nodes show incorrect configurations after upgrade or commit/deploy from Leader

Reported in	Resolved in	Ticket
3.5.1	4.1.1	CRIBL-13863, CRIBL-15507

Problem: When you commit and deploy changes from the Leader, Edge Nodes sometimes fail to automatically restart with the correct configuration changes.

Workaround: Manually restart the Edge Nodes.

Source PQ in Smart mode can trigger excessive memory usage and OOM failures

Reported in	Resolved in	Ticket
3.4.0	4.1.0	CRIBL-13761

Problem: Enabling Source-side Persistent Queues in Smart mode can trigger excessive memory usage, leading to out-of-memory failures and Worker Node instability.

Workaround: If you encounter OOM failures (most likely with high throughput), set PQ to `Always On` mode instead.

Avg Thruput (Bytes Per Second) not displayed for internal logs or metrics

Reported in	Resolved in	Ticket
3.5.1	4.1.0	CRIBL-13530

Problem: No **Avg Thruput (Bytes Per Second)** data is displayed on Cribl Internal logs and metrics Sources' **Charts** tab, nor for these Sources on their attached Routes. This is by design, because Cribl Edge does not perform BPS calculations for internal logs or metrics. But the visual disparity can be confusing.

Workaround: None identified.

New Help drawers open under pinned Help drawers

Reported in	Resolved in	Ticket
3.5.3	4.1.0	CRIBL-12737

Problem: If you've pinned a Help drawer open, it's possible to open additional Help drawers behind the pinned drawer.

Workaround: Close the pinned drawer to see the newly opened drawer.

Cribl Edge/Windows: Upgrading ignores existing mode (etc.) settings

Reported in	Resolved in	Ticket
3.5.0	4.1.0	CRIBL-11467

Problem: When rerunning Windows installers on a system that hosts a previous Cribl version as a managed Edge Node, the installer does not read the distributed mode or other settings. The new version might install as a Leader instance.

Workaround: Run the new version's installer using the same mode and other options you used when installing the preceding version.

Edge Settings mistakenly display Process Count controls

Reported in	Resolved in	Ticket
4.0.0	4.1.1	CRIBL-10402

Problem: Edge Fleets' **Fleet Settings** > **Worker Processes** page incorrectly includes editable **Process Count** and **Minimum Process Count** controls. Each Edge Node is locked to `1` Worker Process, so changes to these fields' values will have no effect.

Workaround: Ignore these two fields.

Chain Function degrades CPU load and throughput

Reported in	Resolved in	Ticket
3.2.2	4.1.0	CRIBL-7445

Problem: Chaining Pipelines via the Chain Function can increase CPU load, and can signficantly slow down data throughput.

Workaround: Consolidate all Functions – per processing scenario – into a single Pipeline.

Resolved in 4.0

The following issues were resolved in version 4.0 and its maintenance releases.

Metrics service unavailable

Reported in	Resolved in	Ticket
4.0.0	4.0.1	CRIBL-13826

Problem: The Monitoring and Fleet pages don't load, due to the systemd-tmpfiles service cleaning some of the Cribl socket files from the `/tmp/` directory.

Workaround: Stop the host system from cleaning the socket files from the `/tmp/cribl-*` directory. For example, on an Amazon EC2 instance, add a new `tmp.conf` file to `/etc/tmp.conf` with the line `X /tmp/cribl-*`. Then restart the tmpfiles cleanup service with: `systemctl restart systemd-tmpfiles-clean.service`. Finally, restart the Cribl server.

Fix: This issue no longer applies to Cribl.Cloud as of 4.0.1. On-prem customers should use the workaround, which will be added to the deployment documentation as a standard practice.

Preview Full > Send Out does not capture events

Reported in	Resolved in	Ticket
3.5.0	4.0.4	CRIBL-14914

Problem: Selecting a Sample Data file's **Preview Full** > **Send out** option captures no events. This bug has been observed when capturing on Destinations.

Workaround: None identified.

Google Cloud Chronicle Destination drops changes in distributed deployments with custom log types

Reported in	Resolved in	Ticket
4.0.0	4.0.3	CRIBL-14453

Problem: A new Google Cloud Chronicle Destination with a custom log type might fail to save changes. This issue affects only distributed deployments.

Workaround: None identified.

Changes to a Lookup table on the Leader don't always propagate to Edge Nodes

Reported in	Resolved in	Ticket
3.5.4	4.0.4	CRIBL-14299

Problem: Modifying a Lookup table on the Leader doesn't always propagate the changes to Edge Nodes, even after clicking **Commit** and **Deploy**.

Workaround: Manually restart the Edge Nodes to refresh Lookup tables.

Users assigned the `owner_all` role cannot perform commits

Reported in	Resolved in	Ticket
4.0.1	4.0.3	CRIBL-14180

Problem: When a user with the `owner_all` role tries to perform a commit, the commit fails with the UI displaying a `Forbidden` modal.

Workaround: Modify the `GroupFull` policy, as follows:

1. As a user with the `owner_all` role, try a `POST /version/commit` API call with a request body of `{ "message": "test: hello" }`. The commit should fail, and the request payload should be `{ message: "test: hello" }`.
2. If `$CRIBL_HOME/local/cribl/policies.yml` does not exist, copy `$CRIBL_HOME/default/cribl/policies.yml` to `$CRIBL_HOME/local/cribl/policies.yml`.
3. If your OS is Linux, run the command `chmod 0744 policies.yml`.
4. In `$CRIBL_HOME/local/cribl/policies.yml`, edit the `GroupFull` policy to match the following:

```
GroupFull:
  args:
    - groupName
  template:
    - PATCH /master/groups/${groupName}/deploy
    - GroupEdit ${groupName}
    - POST /version/commit
    - GET /version
    - GET /version/*
```

5. Now retry the `POST /version/commit` API call, again with a request body of `{ "message": "test: hello" }`. The commit should succeed, and the request payload should be `{ message: "test: hello", "group": "default", "effective": true }`.

Landing page not displaying system metrics from Edge Nodes when teleporting from the Leader

Reported in	Resolved in	Ticket
4.0.1	4.0.3	CRIBL-14169

Problem: When you teleport from a Leader running 4.0.1 or 4.0.2 to an Edge Node that's running 4.0.0 or earlier, the system metrics for the landing page will not populate. This also affects the system metrics shown in the Node drawer for the honeycomb and Node List View pages.

Workaround: Upgrade Edge Nodes to 4.0.2 or higher.

Google Cloud Pub/Sub Source and Destination can break on field validation

Reported in	Resolved in	Ticket
4.0.1	4.0.2	CRIBL-14160

Problem: The Google Cloud Pub/Sub Source and Destination strictly validated entries in the **Topic ID** and **Subscription ID** fields. If you entered a full path (rather than the expected ID substring) in these fields, this strict validation broke the integration and broke the UI.

Workaround: Trim **Topic ID** and **Subscription ID** field values to just the ID.

CrowdStrike FDR Source needs 6-hour visibility timeout

Reported in	Resolved in	Ticket
4.0.2	4.0.3	CRIBL-14067

Problem: With its default **Visibility timeout seconds** setting of `600` (10 minutes), the CrowdStrike FDR Source can accumulate large backlogs when pulling data from Crowdstrike buckets.

Workaround: Set the **Visibility timeout seconds** to `21600` (6 hours), as CrowdStrike recommends. Leave both **Max messages** and **Num receivers** at their default `1` settings.

Amazon S3 Source stops receiving data after upgrade

Reported in	Resolved in	Ticket
4.0.0	4.0.1	CRIBL-14093

Problem: Upgrading to Cribl Edge 4.0.0 can cause the Amazon S3 Source to stop receiving data. This is due to a race condition between (premature) SQS messaging versus the Source's initialization.

Workaround: Skip 4.0.0 (or temporarily roll back to 3.5.4).

QuickConnect-configured Sources' misleading "Enabled" status while disconnected

Reported in	Resolved in	Ticket
4.0.2	4.0.3	CRIBL-14041

Problem: As you configure a new Source from the QuickConnect UI, the Source's **Enabled** slider will initially switch to `Yes`. This is misleading, because the Source is not yet connected to a Destination, so no data can flow.

Workaround: Save the Source in its config drawer. This resets the **Enabled** slider to its accurate `No` status, until you connect the Source to a Destination.

"Unknown config version" error can appear when deploying changes

Reported in	Resolved in	Ticket
4.0.0	4.0.3	CRIBL-13910

Problem: Attempting to commit and deploy a change can trigger errors of the form `Unknown config version: "[hash]"`.

Workaround: On the Leader's host, upgrade the `git` client to 1.9.1 or later.

Teleported Edge Nodes don't display Distributed Settings or Diagnostics Settings

Reported in	Resolved in	Ticket
4.0.0	4.0.1	CRIBL-13868

Problem: When displayed via remote access from the Leader ("teleporting"), a Worker's/Edge Node's **Settings** UI omits the **Distributed Settings** and **Diagnostics** left-nav links. This blocks remote access to features like configuring TLS communications between Edge Node and Leader.

Workaround: Access the Worker's UI directly on its host's port 9000 (or other configured port). As a precondition, you might need to undo any Disable UI Access setting on the parent Fleet.

Chain Function creates extra Pipelines and Functions during initialization

Reported in	Resolved in	Ticket
3.2.0	4.0.1	CRIBL-13860

Problem: Under certain circumstances, when initializing, the Chain Function created multiple unneeded Pipelines and Functions.

Workaround: None identified.

License usage visible only for current day

Reported in	Resolved in	Ticket
4.0.0	4.0.4	CRIBL-13811

Problem: In the listed versions, selecting **Monitoring** > **System** > **License** displays usage only for the current day.

Workaround: None identified.

`Forbidden` error banners mistakenly displayed to non-admin users

Reported in	Resolved in	Ticket
4.0.0	4.0.1	CRIBL-13681

Problem: Non-admin users (with the `reader_all` Role) are mistakenly shown a continuous string of `Forbidden` error banners.

Workaround: Available upon request (documented in the ticket), but cumbersome. Cribl recommends upgrading instead.

Edge/Kubernetes Logs Source repeats logs due to incorrect timestamp

Reported in	Resolved in	Ticket
4.0.0	4.0.3	CRIBL-13900

Problem: The Kubernetes Logs Source is collecting redundant information for containers that don't emit their own timestamps. This Source assumes "current time" when the timestamps are missing, causing it to incorrectly stream older logs during restarts.

Workaround: None identified.

Cribl Edge honeycomb display doesn't render null values

Reported in	Resolved in	Ticket
4.0.0	4.0.1	CRIBL-13595

Problem: When you view Edge Nodes in **Map View**, a honeycomb displays values for each of the metrics you select in the **Measure** drop-down. When a value is null for a particular the Edge Node, the honeycomb should display `null`; instead, the value incorrectly renders as `ul i`.

Workaround: None identified.

Broken in-app doc links

Reported in	Resolved in	Ticket
4.0.0	4.0.1	DOC-100

Problem: The following Sources have broken in-app help links: Kubernetes Logs, Kubernetes Metrics, Windows Event Logs, and Windows Metrics.

Workaround: Use the above links to access these Sources' online documentation.

Spurious “Secret decrypt failed with error" log messages appear after upgrade

Reported in	Resolved in	Ticket
4.0.0	4.0.1	CRIBL-13554

Problem: After upgrading to Cribl Edge 4.0.0, multiple `Secret decrypt failed with error` messages might appear in logs. These are spurious, and you can disregard them.

Workaround: None identified.

Upgrade status shown on Group Upgrade page doesn’t match Global Settings

Reported in	Resolved in	Ticket
4.0.0	4.0.1	CRIBL-13566

Problem: The **Enable automatic upgrades** status shown on the **Settings** > **Group Upgrade** page doesn’t match the slider selection on the **Settings** > **Global Settings** > **Upgrade** tab.

Workaround: View the **Enable automatic upgrades** slider on the **Settings** > **Global Settings** > **Upgrade** tab to verify the current upgrade status.

Sorting by column on Manage > Edge Node page results in blank list

Reported in	Resolved in	Ticket
4.0.0	4.0.1	CRIBL-13505

Problem: When you click a column header on the **Manage** > **Edge Node** page, the list goes blank and rows are not sorted.

Workaround: Click the heading a couple of times to refresh the list.

Git settings don’t immediately populate Commit modals

Reported in	Resolved in	Ticket
4.0.0	4.0.1	CRIBL-13501

Problem: After you save Git settings (such as a default commit message), the new settings do not immediately appear in Commit/Git Changes modals.

Workaround: A hard or soft refresh will close the modal, but when you reopen it, it will reflect your new settings.

Manage Groups/Fleets pages count Packs in Pipeline totals

Reported in	Resolved in	Ticket
3.5.3	4.0.1	CRIBL-12602

Problem: On the **Manage Groups** and **Manage Fleets** pages, the **Pipelines** column overstates the actual number of Pipelines, because it includes Packs in each Fleet's total count.

Workaround: For each Fleet, click the link in its **Pipelines** column to see the Fleet's actual list of Pipelines.

Subfleet search doesn't return parent Fleets

Reported in	Resolved in	Ticket
4.0.0	4.0.1	CRIBL-12465

Problem: On the **Manage Fleets** page's **Fleets** tab, searching for a Subfleet name returns only the Subfleet, without information about its parent Fleets.

Workaround: None identified.

Deleting or modifying default Mapping Rule reclassifies Cribl-managed Workers in Cribl.Cloud as hybrid Workers

Reported in	Resolved in	Ticket
3.2.0	4.0.4	CRIBL-11983

Problem: Cribl.Cloud's UI currently allows deleting or modifying the `default` Mapping Rule. By doing so, an admin can inadvertently reclassify Cribl-managed Workers in Cribl.Cloud as customer-managed (hybrid) Workers. These might not be supported by your Cribl.Cloud plan.

Workaround: If you have an Enterprise plan, create a hybrid Worker Group to manage the resulting hybrid Workers.

Lookup Functions intermittently fail

Reported in	Resolved in	Ticket
3.3.1	4.0.1	CRIBL-9539

Problem: Lookup Functions within some Pipelines were skipped up to ~20% of the time. Restarting Cribl Edge resolves this temporarily, but the failure eventually resurfaces as the new session proceeds.

Workaround: Where a Lookup Function fails, substitute an Eval Function, building a ternary JS expression around a `C.Lookup` method.

Won't Fix

The following issues are not expected to be fixed, either because they are working as designed, because they concern older or unmaintained features, or because they have satisfactory workarounds.

Cribl Copilot for KQL and Pipelines is disabled

Reported in	Resolved in	Ticket
4.8.2	won't fix	AI-797

Problem: With Cribl 4.8.2, the way we manage consent for AI features has changed. As a result, you might have access to Cribl Copilot for chat but not for KQL and Pipelines.

Workaround: Clear the cookies for Cribl.

Examples:

- **Chrome**: Go to **Settings** > **Privacy and security** > **Third-party cookies**, select **See all site data and permissions**, and then find and delete Cribl. Log back in to Cribl.
- **Safari**: Go to **Settings** > **Privacy** > **Manage Website Data**, and select Cribl and delete. Log back in to Cribl.

Cribl can't be upgraded in-product when the `CRIBL_VOLUME_DIR` environment variable is set

Reported in	Resolved in	Ticket
4.8.2	won't fix	CRIBL-28216

Problem: On Cribl Leaders, Cribl Edge, or Cribl Stream Linux instances that have the `CRIBL_VOLUME_DIR` environment variable set, you can't upgrade from 4.8.2 to 4.9.0 using the UI. This affects only upgrades from version 4.8.2. These scenarios are **not** impacted by this issue:

- Upgrading from versions 4.8.1 and older to 4.9.0.
- Installations where `CRIBL_VOLUME_DIR` is not set.
- Windows installations.
- Containers or RPM installs.

Workaround: Upgrade these instances manually. See Upgrading Cribl Edge on Linux (CLI) for instructions.

Edge Leaders running 4.5.1 can't use the Legacy upgrade method to upgrade 4.5.0 Nodes when jobs/tasks are disabled

Reported in	Resolved in	Ticket
4.5.1	won't fix	CRIBL-22801

Problem: Edge Leaders on version 4.5.1 that have Legacy upgrades enabled and **Enable Jobs/Tasks** toggled to `No` cannot upgrade their 4.5.0 Edge Nodes. The job finishes with the task timing out, and the Edge Node is not upgraded.

Workaround: On the 4.5.1 Leader, enable Legacy upgrades **and** toggle **Enable Jobs/Tasks** to `Yes`. Commit and deploy as necessary. The Leader will be able to create a job with the upgrade task and upgrade its Edge Nodes to 4.5.1.

Edge upgrades occasionally fail when Fleets contain a large number of Nodes

Reported in	Resolved in	Ticket
4.2.1	won't fix	CRIBL-20067

Problem: Leader-managed upgrades occasionally fail when a Fleet contains a large number of Nodes.

Workaround: Restart the upgrade operation to upgrade additional Nodes.

Source PQ in Smart mode could cause premature backpressure

Reported in	Resolved in	Ticket
3.4.0	won't fix	CRIBL-13414

Problem: Source-side persistent queuing enabled in `Smart` mode might trigger backpressure before the configured maximum queue size is reached. For TCP Sources, this will send backpressure back to the sender. For Sources using the UDP protocol, it will cause events to be dropped.

Workaround: None identified.

Default AppScope Config file can't be restored when in use

Reported in	Resolved in	Ticket
4.0.0	won't fix	CRIBL-13400

Problem: When the default AppScope Config file is assigned to an AppScope Source filter, you can't restore it to its default settings. You'll see an error message that the AppScope Config file is currently in use.

Workaround: On the AppScope Source's **AppScope Filter Settings** tab, delete the default AppScope Config from the **Allowlist**. Then, edit the AppScope Config Knowledge object to restore its default settings. You can now add this AppScope Config, with its restored settings, back to the AppScope Source's Allowlist.

`cidr-matcher` does not support IPv6 addresses

Reported in	Resolved in	Ticket
3.0.0	won't fix	CRIBL-11767

Problem: The version of `cidr-matcher` supported by Cribl only supports IPv4 CIDR matching.

Workaround: None identified.

Cribl.Cloud login page distorted on iPad

Reported in	Resolved in	Ticket
3.2.2	won't fix	SAAS-1141

Problem: On certain iPads, we've seen the Cribl.Cloud login page's left text column repeated twice more across the display. These unintended overlaps prevent you from selecting (or tabbing to) the **Log in with Google** button.

Workaround: If you encounter this, the only current workarounds are to either use Google SSO on a desktop browser, or else use a different login method.

16.2. Get Product Help

This page has moved.

Please see Get Product Help in the Cribl Product Suite section of Cribl Docs.

16.3. Common Challenges On The Edge

Here are some common issues that users have encountered when using Cribl Edge, along with guidance to resolve them.

Licensing and Quotas Confusion

Managed Edge Nodes use their Leader's licensed ingestion quota, but only for Sources that are enabled and configured. Data from some Sources like system state that collect data locally are not counted against the license. Data sent to Cribl Stream via TCP/HTTP is not counted against the license. For details see, Licensing.

Disable SNI Routing for Firewalls/Web Proxies

There is a setting in `cribl.yml` to help you manage Server Name Indication (SNI) routing, which can be helpful if you're encountering difficulties with proxies or firewalls.

If you have a proxy or firewall deployed between your Edge Nodes and the Leader, this new setting allows you to disable SNI routing for specific Fleets.

To disable SNI routing for a Fleet: Navigate to **Fleet Settings** > **General Settings** > **SNI** > **Disable SNI Routing** option. **Commit** and **Deploy** and then proceed with the upgrade. You must enable this setting for every Fleet that is behind a web proxy or firewall.

Web UI Errors

While we hope you have an error-free experience, we want to prepare you in case you run into issues. Here are some errors you might encounter in Edge, and how to resolve them:

Error: "No workers registered"

Cause: Occurs 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.

Recommendation: There are two things you can try:

- 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.

Leader and Edge Node (In)Compatibility

Leaders on v.4.2.x are compatible with Edge Nodes on v.4.1.2 and later. Due to a security update, Edge Nodes running on v.4.0.4 cannot receive configurations from v.4.2.x Leaders. For details, see Leader and Edge Nodes Compatibility.

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:

Linux

```
# cat /opt/cribl/local/cribl/auth/676f6174733432.dat

{"it":1570724418,"phf":0,"guid":"48f7b21a-0c03-45e0-a699-01e0b7a1e061"}
```

Windows

- PowerShell

```
PS C:\ProgramData\Cribl\local\cribl\auth> type 676f6174733432.dat

{"it":12345678901,"phf":0,"guid":"ff983143-1f93-4b61-a442-e5db8ccb1f50","lics":[
```

- CMD

```
type C:\ProgramData\Cribl\local\cribl\auth> type 676f6174733432.dat

{"it":12345678901,"phf":0,"guid":"ff983143-1f93-4b61-a442-e5db8ccb1f50","lics":[
```

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 regenerate on the next run.

Running Edge (Linux) as Non-root 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. For guidelines, see Running Edge as an Unprivileged User.

Debugging Cribl Edge on Windows

Common issues users encounter when deploying Cribl Edge on Windows include:

Using a Pre-4.3.0 MSI Installer

MSIs earlier than v.4.3.0 install the app so it belongs to the user running the installer. If you later try to upgrade it as another user, things go awry — resulting in a rather confusing mess with a seemingly random mix of the old and new package contents on disk.

Moving forward, we recommend adding `ALLUSERS=1` to the `msiexec` command when installing or upgrading pre-4.3.0 versions 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

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

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.

CPU/Memory Issues on Windows

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

- 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 that collector if you see that process consuming excessive CPU.
- If you are interested in collecting logs only, disable the Windows Metrics 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 Windows Event Forwarder and send them to Stream instead of using Windows Event Log Source in Cribl Edge.

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.

To tell Windows Defender to exclude the Cribl Edge process from its security scan, you can add this command to the appropriate Windows Edge Nodes:

```
Add-MpPreference -ExclusionProcess "C:\Program Files\Cribl\bin\nssm.exe"
```

Local Files When Uninstalling Windows

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 these folders from the filesystem.

Edge via Kubernetes

Common mistakes users make when deploying via Kubernetes include:

Failing Health Checks

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`). If you have such nodes, 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.

Not Setting the Environment Variables for Invalid Certifications

If your Kubernetes deployment doesn't use valid certificates, 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 AppScope metadata) will tolerate invalid (expired, self-signed, etc.) TLS certificates when connecting to the Kubernetes APIs. If this environment variable is not defined or set to 1 and the certificate validation fails, all connection attempts to the Kubernetes API will be rejected.

Not Authorizing the Cribl Edge Pod Service Account

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. For details, see RBAC.

Not Deploying Cribl Edge as a Daemonset

The Kubernetes Sources (Events, Logs, Metrics) 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 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.

For the Kubernetes Events Source to work as designed, we recommend deploying Cribl Edge as a Daemonset. For details, see Deploying via Kubernetes.

Which Destination to Use for 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 the license.

Unsupported Files on the Journal Files Source

This Source does not support the new `COMPACT` binary file format. The logs will contain an error message when the parser encounters this type of file. (A fix is planned for v.4.3.1.). Sample error message when enabling the Journal Files Source:

```
{ [- ]
channel: input: in_journal_local cid: WO err: { [-]
message: Cannot read properties of undefined (reading 'data')
stack: TypeError: Cannot read properties of undefined (reading 'data')
at w. readEntry (/opt/cribl-edge/bin/crib1.js:14:18884713)
at w. readEvents (/opt/cribl-edge/bin/crib1.js:14:18885930)
at w._transform (/opt/crib1-edge/bin/crib1.js:14:18886631)
at w. Transform. _write (node: internal/streams/transform:205:23) at writeOrBuffer (
}
level: warn
message: Error while stopping journal collector time: 2023-08-15T21:04:51.929Z
Show as raw text
host = ip-10-99-99-38.us-west-2.compute. internal
source = cribl: sourcetype = cribledge
```

Also, this Source does not currently 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. Here is an example of the error message:

```
Error reading journald event, dropping offset=31678152 evt="Mar 02 21:29:54 goats-xps15 multipassd: undefined"
```

File Monitor Source

Common mistakes users make when exploring the File Monitor Source include the following:

Using an Editor

Do not test the Source by editing and saving files in a directory using a text editor (vi or Notepad). Under the hood, these editors operate on a temporary file that they move into place when saved. As a result, the File Monitor 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.

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":"..." }`

Misunderstanding the Filename Allowlist

The Filename allowlist in the Source applies the wildcard patterns to the entire path of the discovered file, not just the file name.

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`.

For details, see Using Allowlists Effectively.

Not configuring the Hash Length to Avoid Common Headers

It's common for a batch of CSV files or IIS logs to have identical first lines listing columns that appear in subsequent lines. 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. For details, see Configuring Hash Lengths.

Allowing Race Conditions

The hashing logic isn't instantaneous, so it's possible for the common header problem described above to be confusing. 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 head-hash matches an existing entry in the state-store.

Testing Multiple Copies of the Same File

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. You can't copy the same file into the Source's path over and over and expect them to be collected repeatedly.

Pointing Multiple File Monitor Sources to 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.

Testing with Small Files

The 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. There is some exception logic for dealing with files smaller than the

hash sizes that cannot be made perfect. If you try to test with small files (smaller than 256 bytes), you will be working with the fallback logic and not the logic used for normal logs.

Panicking over Variable File Sizes

The File Monitor Source can handle a variety of file sizes without “starving” the smaller ones. In Cribl Edge, multiple workflows can be multiplexed using the event loop, even though it runs on a single core.

Not Testing with Event Breakers

To make sure that incoming files are splitting 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 details, see Timestamp Settings.

16.4. 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, the bundle excludes all `.crt`, `.pem`, `.cer`, and `.key` files 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 `Yes`.
- **Include metrics**: Includes metrics in the bundle. Defaults to `Yes`.
- **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 `Yes`.
- **Latest number of jobs to include**: The number of jobs (and all tasks) the bundle will include. Defaults to empty, which means unlimited jobs.
- **Include install logs**: Includes Windows MSI logs, only available on Windows deployments of Cribl Edge. Defaults to `No`.
    - 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. Disabling **Include metrics** can reduce the size of the diag bundle, where necessary.

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** to `Yes`. 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 UI Access (teleporting) setting:

1. In the Cribl Edge sidebar, select **Fleets**.
2. Make sure the relevant Fleet's **UI Access** 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
  [-t <maxIncludeJobs>] - Latest number of jobs to include in bundle
  [-M]                  - Exclude metrics from bundle
  [-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 create

## 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
```

For complete command syntax, see diag.

Include CPU Profiles

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 the setting to `Yes`. 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
```

3. Run the `diag heapsnapshot` command with the PID you noted in step 1.

- For Linux:

```
./cribl diag heapsnapshot -p <pid>
```

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"
```

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
```

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>
```

Cribl will capture the memory snapshot in `$CRIBL_HOME/diag/`.

Example output:

```
Heap-1690906031148-2458958.heapsnapshot
```