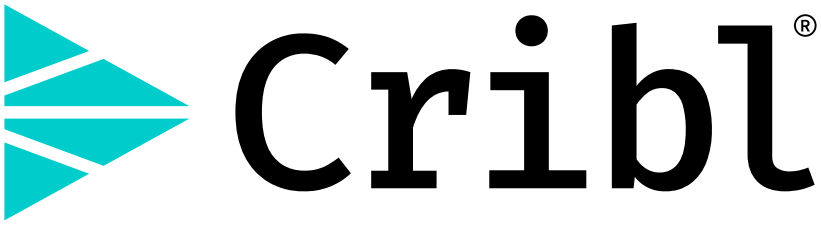


Cribl Edge Documentation Manual

Version: v4.2

Generated: 2023-08-16 13:46:30

1. Introduction 9
1.1. About Cribl Edge 9
1.2. Basic Concepts 11
1.3. Exploring Cribl Edge on Linux 16
1.4. Exploring Cribl Edge on Windows 45
1.5. QuickConnect 63
2. Cribl.Cloud 70
2.1. Launch Guide 70
2.2. Cloud SSO Setup 95
2.2.1. Cribl.Cloud SSO Setup 95
2.2.2. Common SSO Setup Steps 98
2.2.3. OIDC/Okta Setup Example 100
2.2.4. SAML/Azure AD Setup Examples 108
2.2.5. SAML/Okta Setup Examples 111
2.2.6. Final SSO Steps & Troubleshooting 122
3. Deploying Edge 124
3.1. Deployment Planning 124
3.2. Installing Cribl Edge on Linux 130
3.2.1. Enabling Start on Boot 134
3.2.2. Running Edge as an Unprivileged User 138
3.2.3. Anti-Virus Exceptions 140
3.2.4. System Proxy Configuration 141
3.3. Installing Cribl Edge on Windows 146
3.4. Running in a Docker Container 158
3.5. Deploying via Kubernetes 161
4. Fleet Management 168
4.1. Fleets 171
4.2. Managing Edge Nodes 182
4.3. Setting Up Leader and Edge Nodes 189
4.4. Leader High Availability/Failover 194
5. Cribl Edge Better Practices 201
5.1. Monitoring your Infrastructure with Cribl Edge 201
5.2. Fleets Hierarchy and Design 207
5.3. Kubernetes Observability Using Cribl Edge 211
5.4. Windows Observability Using Cribl Edge 223
6. Administering 235

6.1. Upgrading 235
6.2. Uninstalling Cribl Edge from Linux 245
6.3. Uninstalling Cribl Edge from Windows 246
6.4. Licensing 247
6.5. About Load Balancing 254
6.6. Persistent Queues 256
6.7. Fleet Settings 265
6.8. Monitoring and Notifications 267
6.8.1. Monitoring Health and Metrics 267
6.8.1.1. Internal Metrics 281
6.8.1.2. Internal Logs 290
6.8.2. Notifications 295
6.8.2.1. Configuring Targets 304
6.9. Scripts 309
7. Access Management 311
7.1. Authentication 312
7.2. Members and Permissions 319
7.3. Local Users 326
7.4. Roles 328
7.5. Using ACLs to Allow Cribl Edge to Read Files 338
8. Securing 341
8.1. Securing the Leader's Auth Token 341
8.2. Securing Cribl Edge (TLS/SSL, Certs, Keys) 343
8.2.1. KMS Configuration 350
8.3. Securing Communications 355
9. Sources 365
9.1. System 369
9.1.1. Exec 369
9.1.2. File Monitor 372
9.1.3. Journal Files 380
9.1.4. Kubernetes Events 384
9.1.5. Kubernetes Logs 390
9.1.6. Kubernetes Metrics 394
9.1.7. System Metrics 399
9.1.8. System State 404
9.2. Internal 415
9.2.1. Cribl HTTP 415

9.2.2. Cribl TCP 422
9.2.3. Cribl Stream (Deprecated) 428
9.2.4. Cribl Internal 434
9.2.5. Datagen 440
9.3. Amazon 442
9.3.1. Amazon Kinesis Firehose 442
9.4. Prometheus 448
9.4.1. Prometheus Edge Scraper 448
9.4.2. Prometheus Remote Write 456
9.4.3. Grafana 462
9.4.4. Loki 468
9.5. Splunk 474
9.5.1. Splunk HEC 474
9.5.2. Splunk TCP 484
9.6. Windows 491
9.6.1. Windows Event Forwarder 491
9.6.2. Windows Event Logs 506
9.6.3. Windows Metrics 510
9.7. AppScope 514
9.8. Datadog Agent 522
9.9. Elasticsearch API 531
9.10. HTTP/S (Bulk API) 540
9.11. Raw HTTP/S 548
9.12. Metrics 554
9.13. OpenTelemetry (OTel) 560
9.14. SNMP Trap 567
9.15. Syslog 571
9.16. TCP JSON 578
9.17. TCP (Raw) 585
9.18. UDP (Raw) 591
10. Destinations 595
10.1. Internal 600
10.1.1. Cribl HTTP 600
10.1.2. Cribl TCP 606
10.1.3. Cribl Stream (Deprecated) 613
10.1.4. Default 618
10.1.5. DevNull 620

10.1.6. Output Router 621
10.2. Amazon 624
10.2.1. Amazon CloudWatch Logs 624
10.2.2. Amazon Kinesis Streams 629
10.2.3. Amazon S3 Compatible Stores 634
10.2.4. Amazon SQS 641
10.2.5. Amazon Security Lake 646
10.3. Azure 653
10.3.1. Azure Blob Storage 653
10.3.2. Azure Event Hubs 659
10.3.3. Azure Monitor Logs 665
10.3.4. Azure Sentinel 670
10.4. Google Cloud 676
10.4.1. Google Chronicle 676
10.4.2. Google Cloud Logging 681
10.4.3. Google Cloud Pub/Sub 686
10.4.4. Google Cloud Storage 690
10.5. Kafka 695
10.5.1. Kafka 695
10.5.2. Confluent Cloud 701
10.5.3. Amazon MSK 707
10.6. Metrics 713
10.6.1. Graphite 713
10.6.2. StatsD 716
10.6.3. StatsD Extended 719
10.7. New Relic Ingest 722
10.7.1. New Relic Events 722
10.7.2. New Relic Logs & Metrics 727
10.8. Prometheus 732
10.8.1. Prometheus 732
10.8.2. Grafana Cloud 737
10.8.3. Loki 744
10.9. Splunk 749
10.9.1. Splunk HEC 749
10.9.2. Splunk Single Instance 756
10.9.3. Splunk Load Balanced 762
10.10. CrowdStrike Falcon LogScale 771

10.11. Datadog 776
10.12. DataSet 781
10.13. Elasticsearch 786
10.14. Filesystem/NFS 792
10.15. Honeycomb 797
10.16. InfluxDB 801
10.17. MinIO 805
10.18. OpenTelemetry (OTel) 812
10.19. SignalFx 818
10.20. SNMP Trap 822
10.21. Sumo Logic 824
10.22. Syslog 828
10.23. TCP JSON 842
10.24. Wavefront 848
10.25. Webhook 852
11. Working with Data 861
11.1. Event Model 861
11.2. Event Processing Order 863
11.3. Routes 865
11.4. Pipelines 873
11.5. Packs 881
11.5.1. Packs Publication Standards 907
11.6. Using Datagens 915
11.7. Data Preview 921
11.8. Data Onboarding 938
12. Functions 942
12.1. Auto Timestamp 948
12.2. Aggregations 953
12.3. CEF Serializer 963
12.4. Chain 965
12.5. Clone 967
12.6. Code 969
12.7. Comment 972
12.8. DNS Lookup 974
12.9. Drop 980
12.10. Dynamic Sampling 982
12.11. Eval 985
12.12. Event Breaker 989

12.13. Flatten 992
12.14. GeoIP 994
12.15. Grok 997
12.16. JSON Unroll 999
12.17. Lookup 1001
12.18. Mask 1007
12.19. Numerify 1014
12.20. Parser 1016
12.21. Publish Metrics 1028
12.22. Redis 1034
12.23. Regex Extract 1044
12.24. Regex Filter 1049
12.25. Rename 1050
12.26. Rollup Metrics 1053
12.27. Sampling 1055
12.28. Serialize 1057
12.29. Suppress 1060
12.30. Tee 1063
12.31. Trim Timestamp 1066
12.32. Unroll 1069
12.33. XML Unroll 1071
12.34. Prometheus Publisher (Deprecated) 1074
12.35. Reverse DNS (deprecated) 1077
13. Knowledge Objects 1079
13.1. Lookups Library 1080
13.2. Event Breakers 1084
13.3. Parsers Library 1098
13.4. Global Variables Library 1100
13.5. Regex Library 1102
13.6. Grok Patterns Library 1105
13.7. Schemas Library 1107
13.8. Parquet Schemas 1110
13.9. AppScope Configs 1118
14. Reference 1126
14.1. API Docs 1126
14.2. CLI Reference 1135
14.3. Config Files 1152
14.3.1. cribl.yml 1156

14.3.2. breakers.yml 1162
14.3.3. certificates.yml 1163
14.3.4. groups.yml 1164
14.3.5. inputs.yml 1165
14.3.6. instance.yml 1246
14.3.7. jobs.yml 1247
14.3.8. job-limits.yml 1268
14.3.9. licenses.yml 1269
14.3.10. limits.yml 1270
14.3.11. logger.yml 1271
14.3.12. mappings.yml 1272
14.3.13. messages.yml 1273
14.3.14. outputs.yml 1274
14.3.15. parsers.yml 1345
14.3.16. policies.yml 1346
14.3.17. regexes.yml 1349
14.3.18. roles.yml 1350
14.3.19. samples.yml 1351
14.3.20. schemas.yml 1352
14.3.21. scripts.yml 1353
14.3.22. vars.yml 1354
14.4. Cribl Expressions 1355
14.5. Cribl Expression Syntax 1368
14.6. Environment Variables 1371
14.7. Linux System Metrics Details 1377
14.8. Windows System Metrics Details 1391
15. Techniques & Tips 1422
15.1. Cribl Edge to Cribl Stream 1422
15.2. Cribl Edge as a Sidecar Container in AWS ECS 1428
15.3. Amazon Security Lake Integration 1433
15.4. SSO with Okta and OIDC 1441
15.5. SSO with Okta and SAML 1457
16. Troubleshooting 1471
16.1. Known Issues 1471

1. Introduction

1.1. About Cribl Edge

All the docs to goat you started with Cribl Edge.

Download all docs as a PDF – v.4.2.1 | Download Binary | Edge on Cribl.Cloud

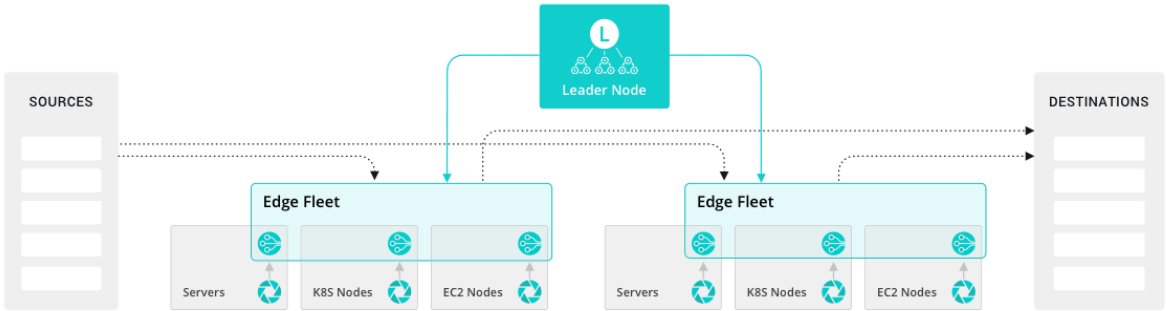
- Questions not answered here? We'd love to help you. Meet us in #Cribl Community Slack – sign up here.

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



Who Is Cribl Edge For?

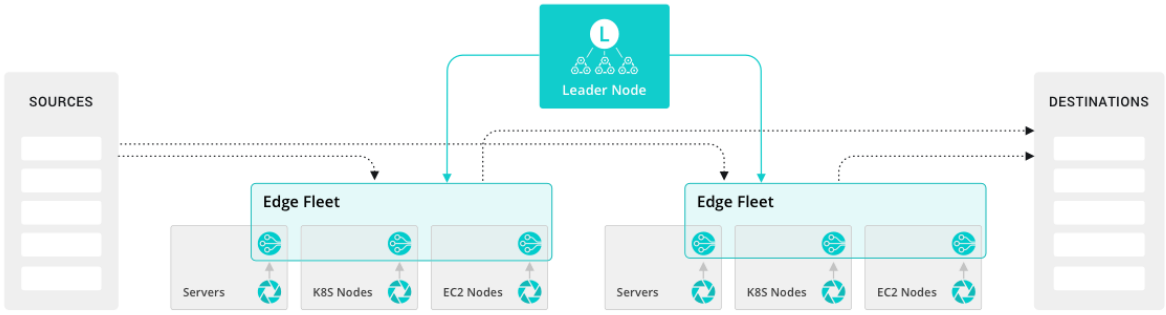
Cribl Edge is built for ITOps and SecOps Engineers and users of operational and security intelligence products and services who want to simplify the process of collecting and analyzing data from a host.

> Cribl University offers two courses that provide a good overview of Cribl Edge: The What & Why of Cribl Edge and Reviewing Key Concepts.To follow the direct course link, first log into your Cribl University account. (To create an account, click 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

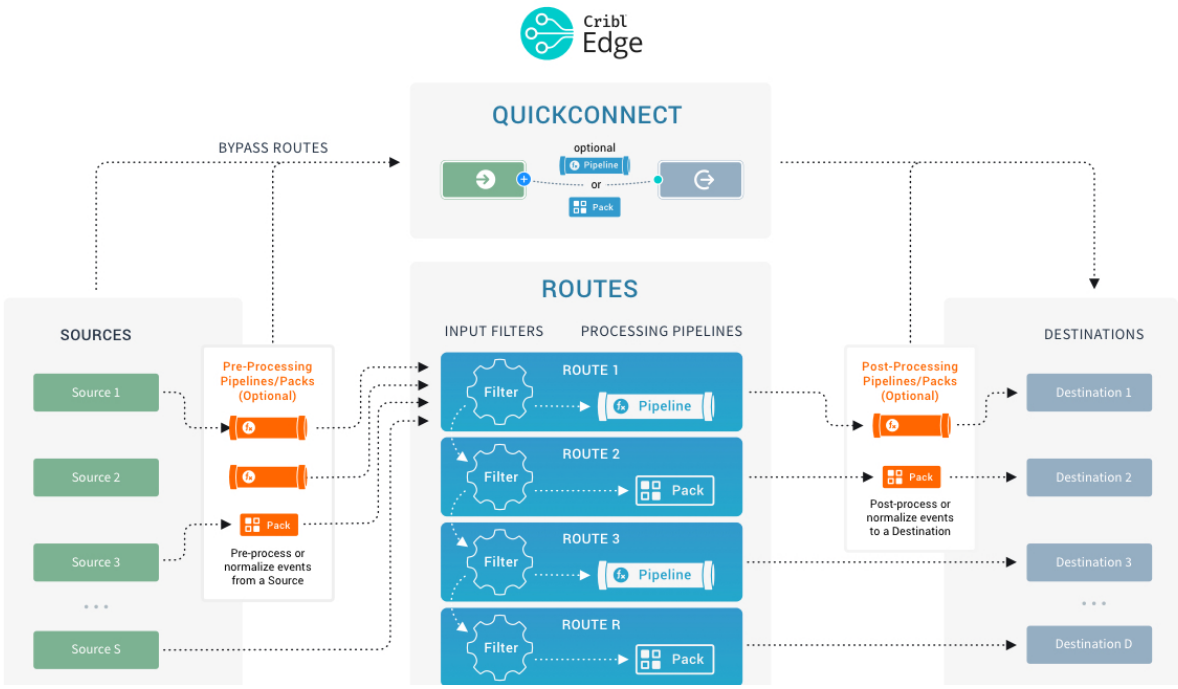
Notable features and concepts to get a fundamental understanding of Cribl Edge

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

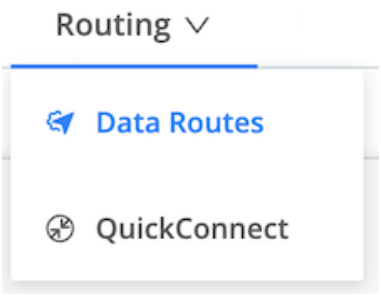
Sources are configurations that enable Edge to collect data from the local machine (logs, metrics, etc.) or to receive data from remote senders (TCP, Syslog etc.).

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.

You use the top nav's **Routing** menu to toggle between the **Data Routes** and **QuickConnect** interfaces.



Select Data Routes to use the Routing table, or QuickConnect to bypass Routes

Routes

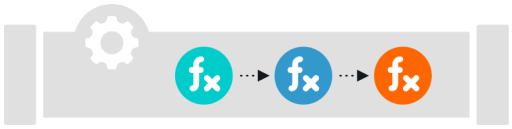
Routes evaluate incoming events against **filter** expressions to find the appropriate Pipeline to send them to. Routes are **evaluated in order**. Each Route can be associated **with only one** Pipeline and one output (configured as a Edge **Destination**).

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.

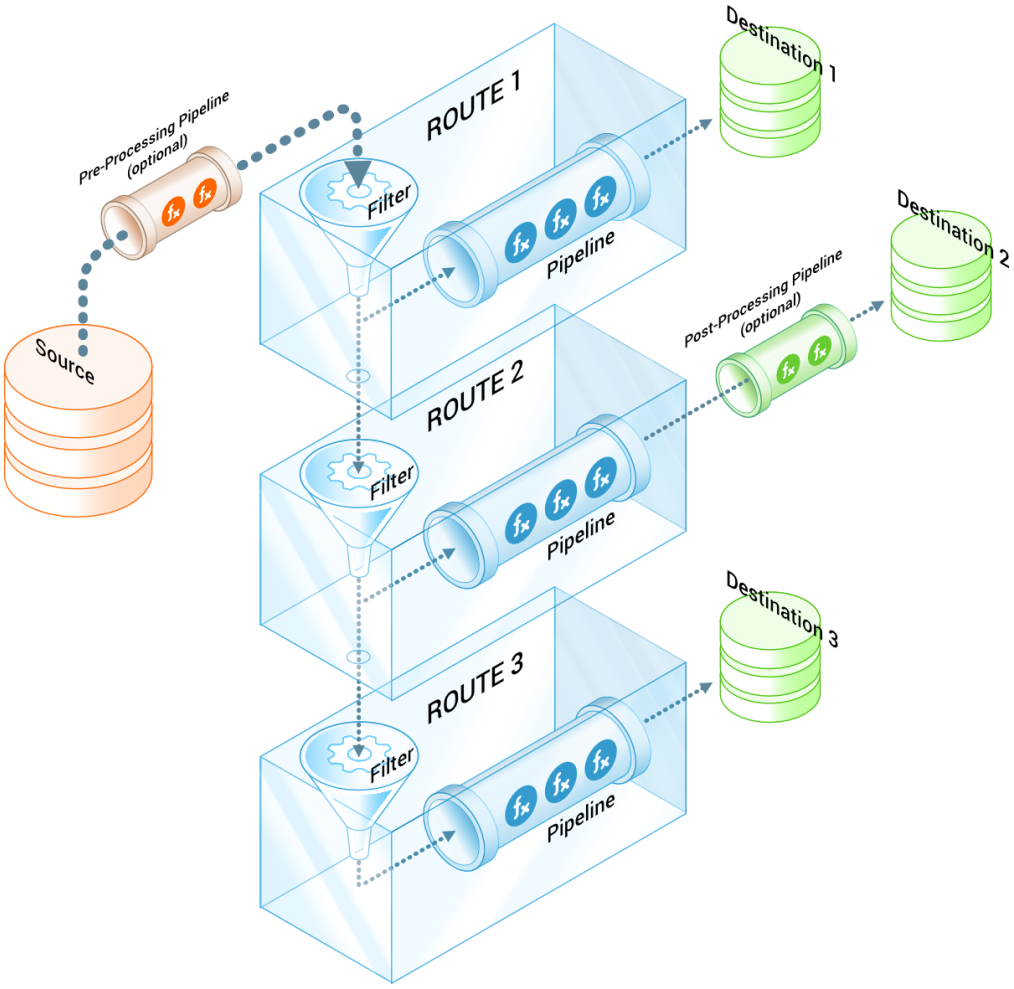
However, if you disable the Route's `Final` flag, one or more event **clones** will be sent down the associated Pipeline, while the original event continues down Edge's Routing table to be evaluated against other configured Routes. This is very useful in cases where the same set of events needs to be 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.

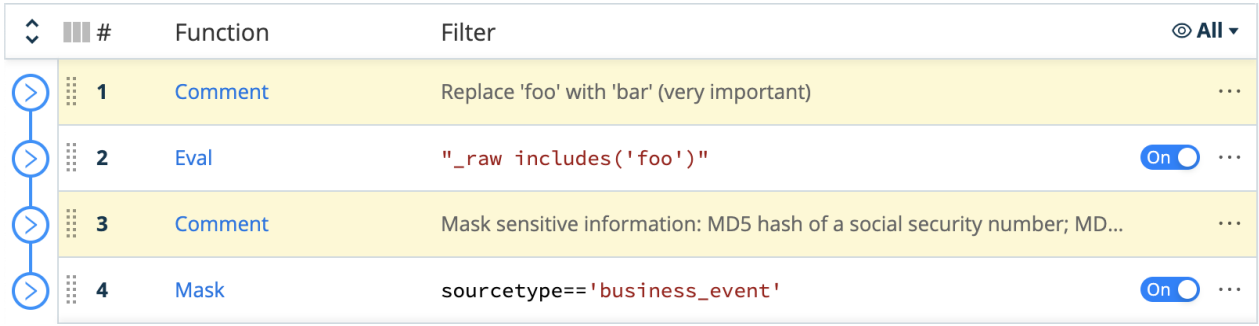


Pipelines categorized by position

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

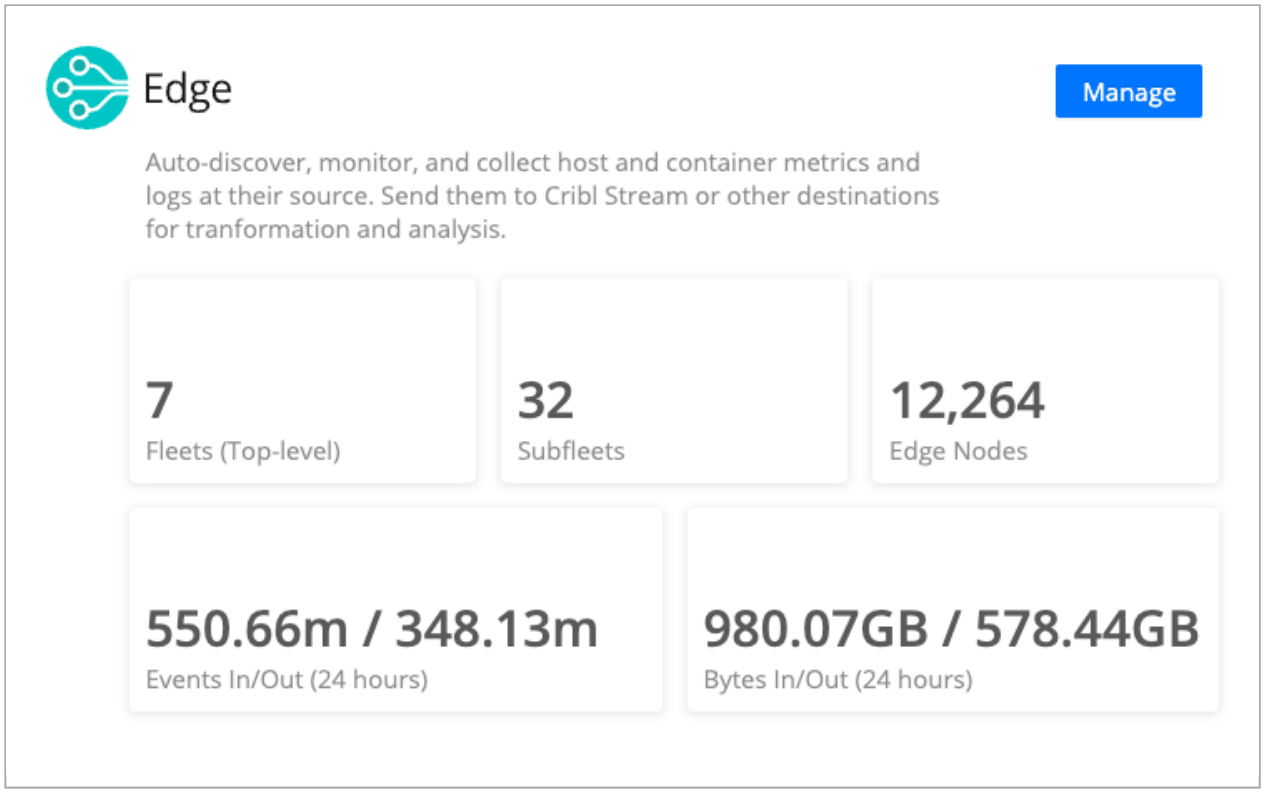
Functions process each event that passes through them. To help improve performance, Functions can optionally be configured with filters, to limit their processing scope to matching events only. For more details, see Functions.

1.3. Exploring 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.

Accessing Cribl Edge

When you first log into Cribl Stream/Edge (single-instance or distributed), you'll see tiles that prompt you to choose between ~~two roads diverging in a yellow wood~~ the Stream versus Edge UIs. The Edge tile displays basic configuration details, including the number of Fleets, Subfleets, Edge Nodes, and events and bytes over time. Click **Manage** to start.

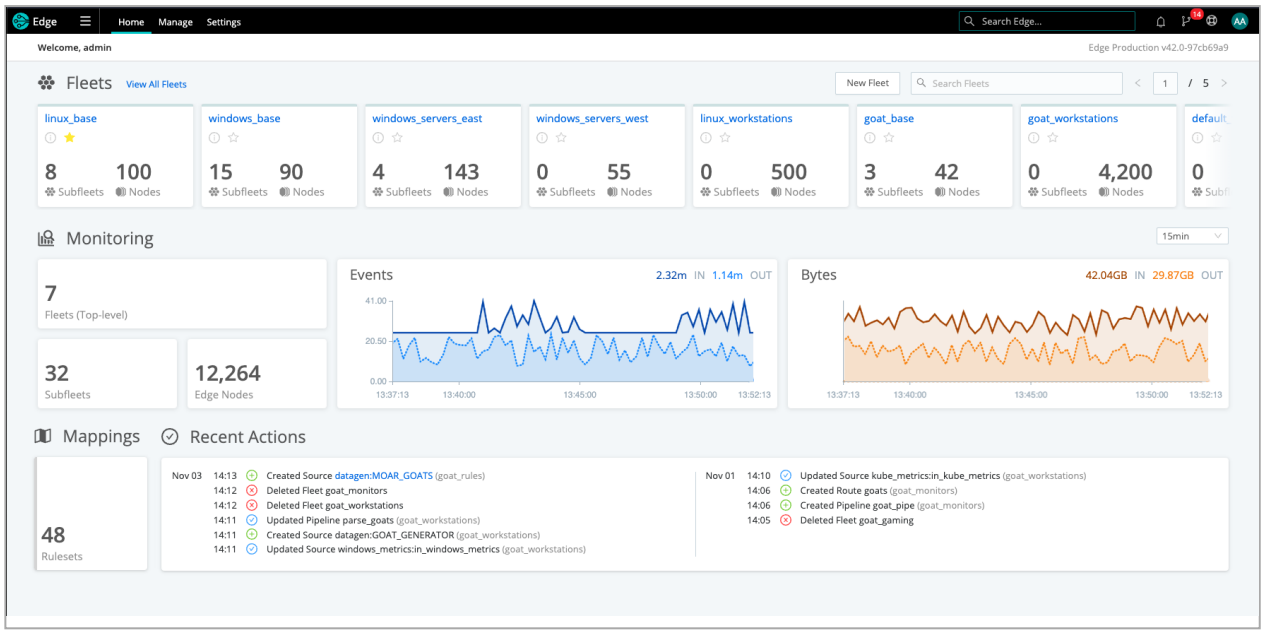


Manage Edge

Fleets Overview

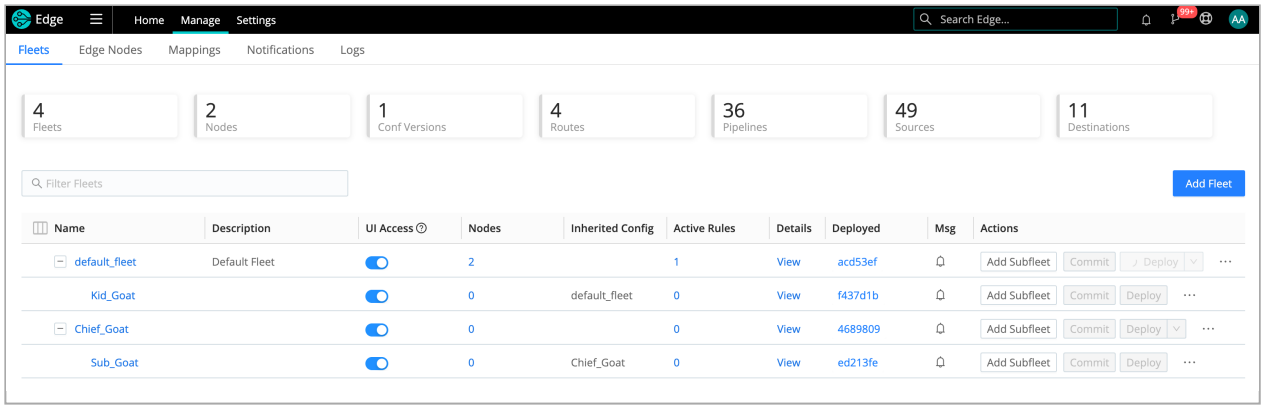
On the Cribl Edge **Home** tab, you can access your configured Fleets and a summary of your Cribl Edge environment, highlighting the aggregate data for all Fleets, Subfleets, Edge Nodes, and Mappings. The charts

display information about traffic in and out of the system.



Cribl Edge Homepage

Select **Manage** from the top nav to view the **Fleets Landing** page. Here you can access the tabs for more information about your **Fleets** (and **Subfleets**), **Edge Nodes**, **Mappings**, **Notifications**, and **Logs**.

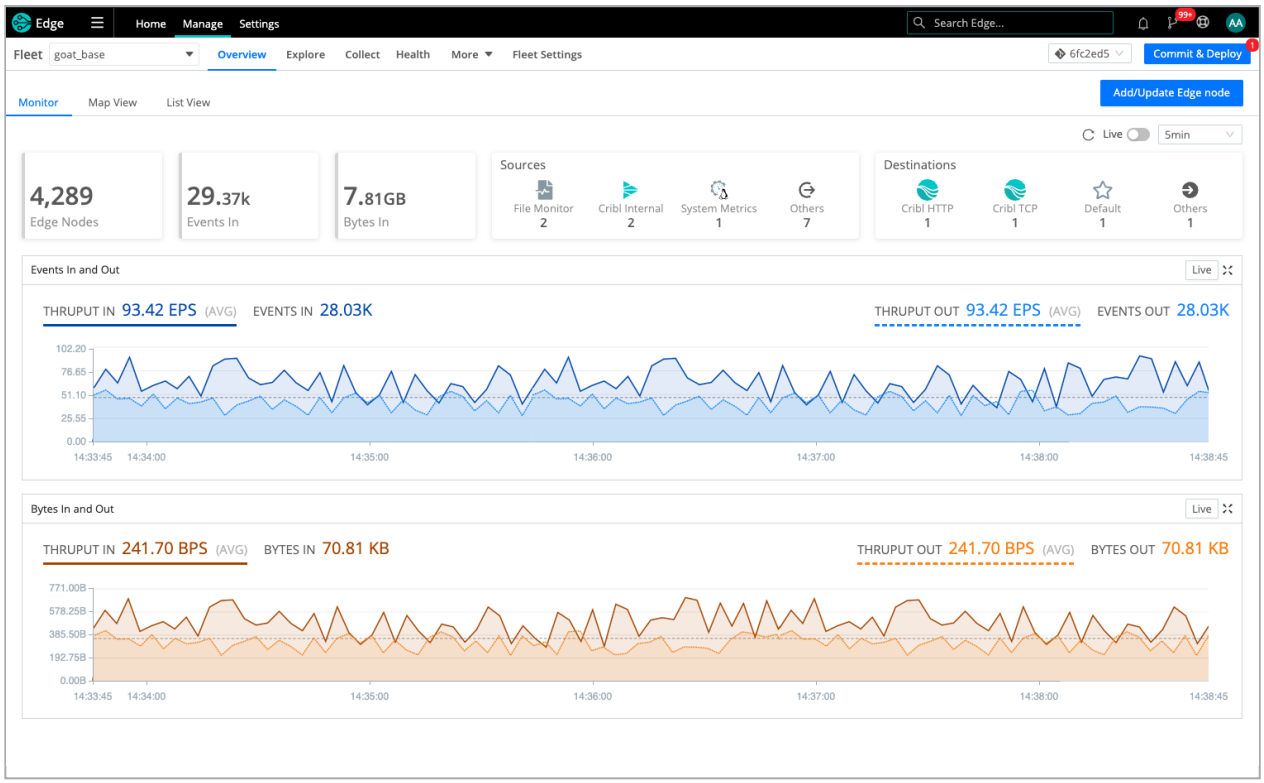


Manage Fleets

The **Manage Fleets** page gives you access to more information about your **Fleets** (and **Subfleets**), **Edge Nodes**, **Mappings**, **Notifications**, and **Logs**.

You can click a **Fleet** link to isolate individual Fleets, or use the **Search** bar to locate your Fleet.

Fleet Landing Page



Fleet Landing page

The Fleet's landing page highlights information about your configured Edge Nodes. The following information is displayed across the top.

Number of Edge Nodes: How many Edge Nodes are configured in this Fleet.

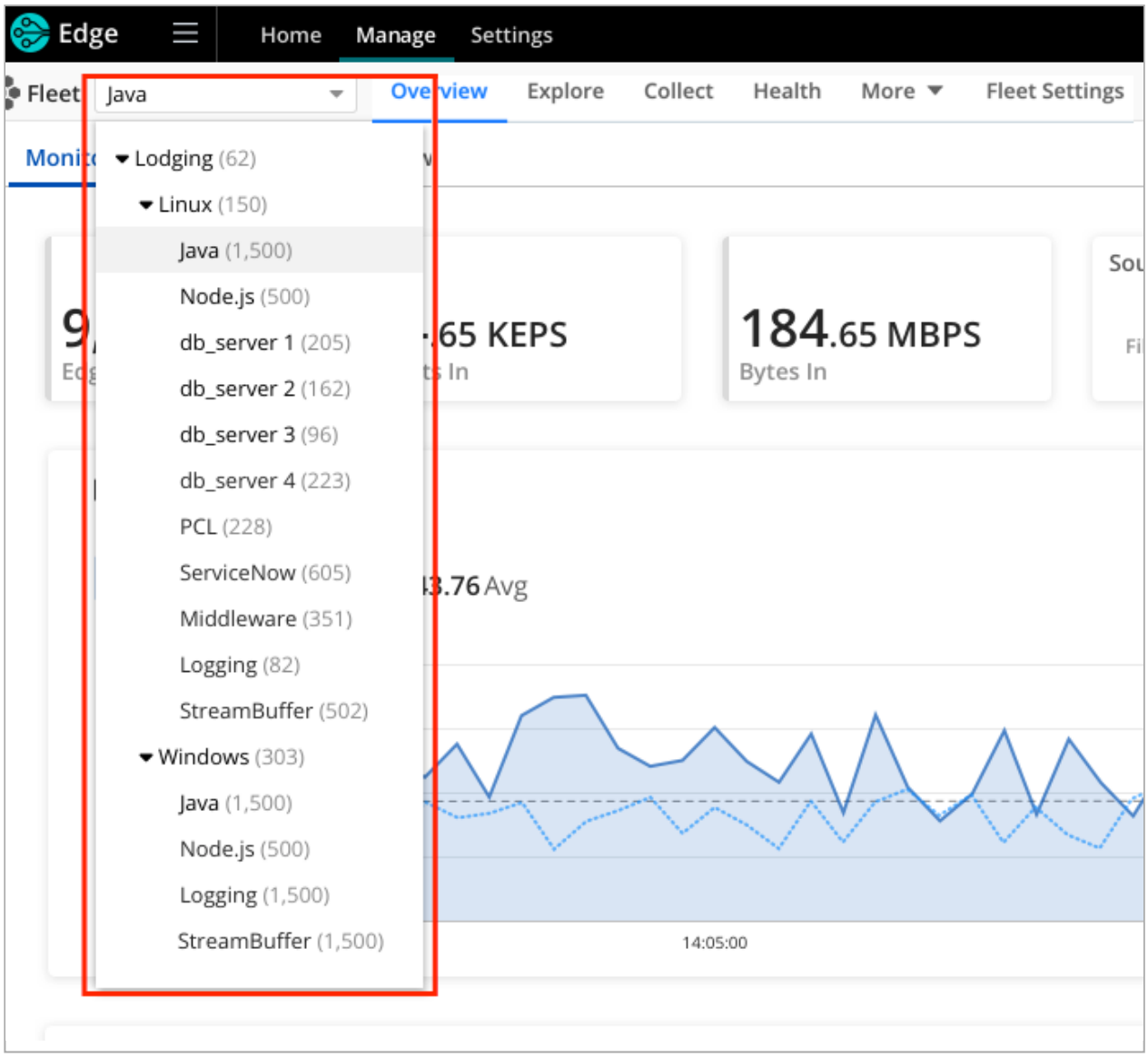
Events In: Total number of Events in the last 5 minutes of data collected. You can change the display's granularity from the default `last 5 min`, selecting a variety of time ranges from 1 min up to 1 day. (The latter covers the preceding 24 hours, and this maximum window is not configurable.)

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.

Select the **Fleet** dropdown (top right) to see a hierarchical list of all your Fleets and Subfleets.



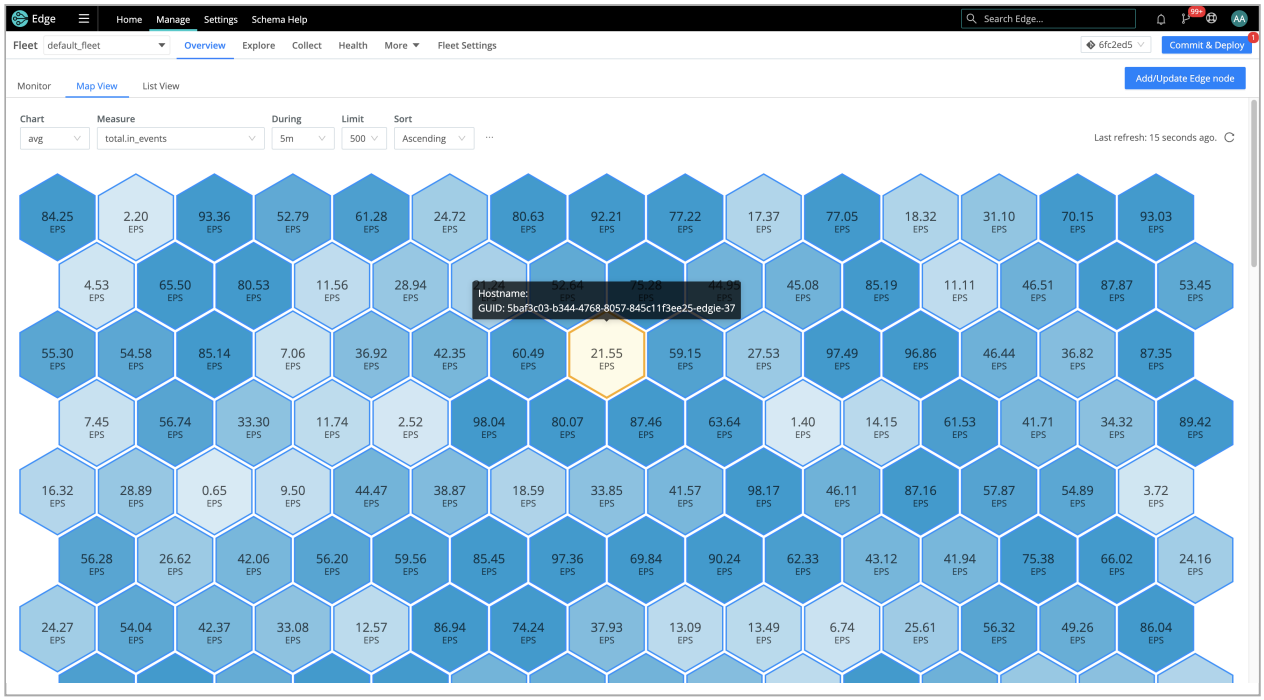
Fleets and SubFleets list

Fleet Map View

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.

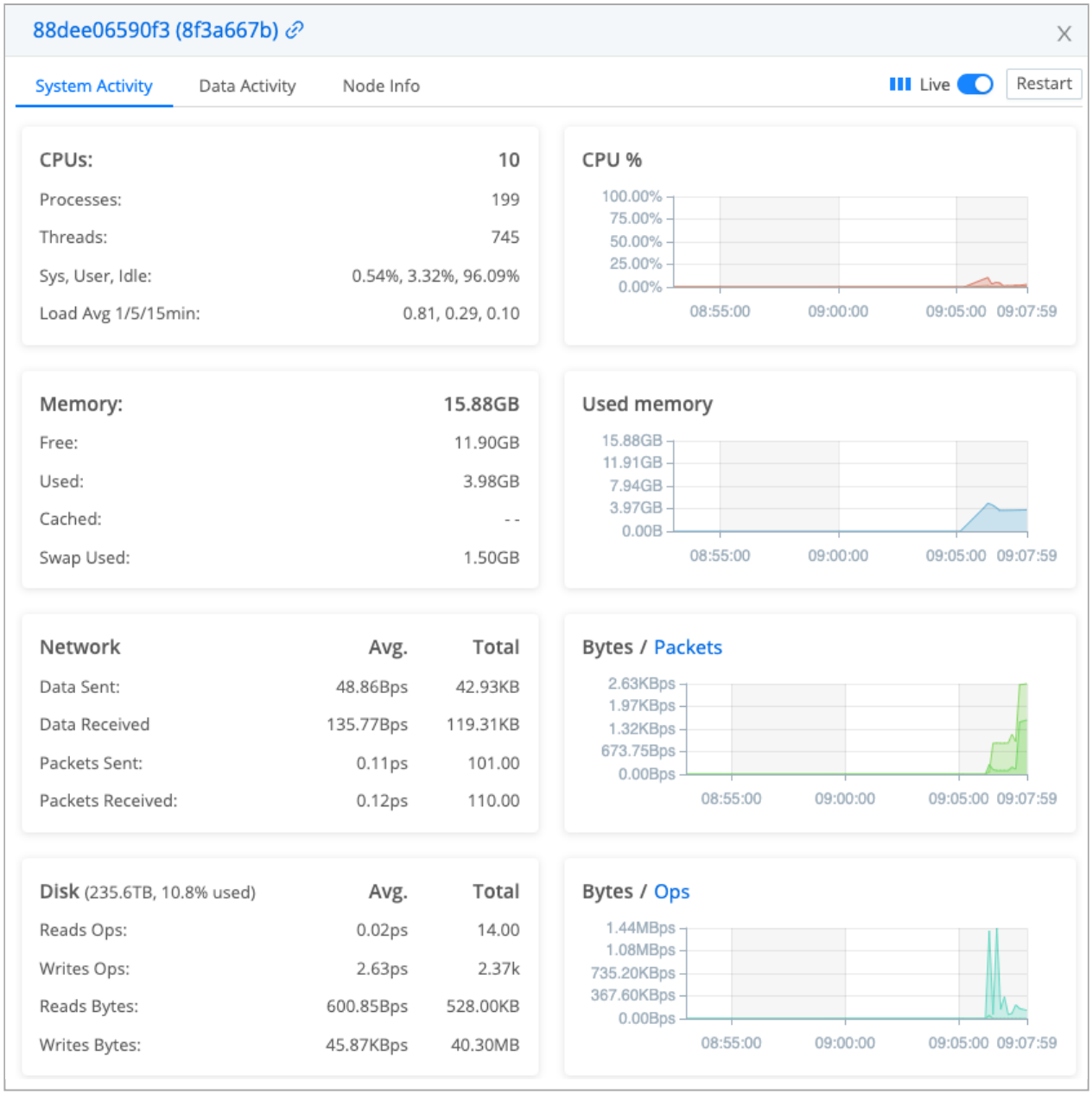
The metrics that appear in the **Measure** list depend on the option selected in **Fleet Settings** > **Limits** > **Metrics** under **Metrics to send from Edge Nodes**. The metrics `total.in_bytes`, `total.in_events`, `total.out_bytes`, and `total.out_events` always appear in the list; the display of 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 Edge Node's **GUID**.



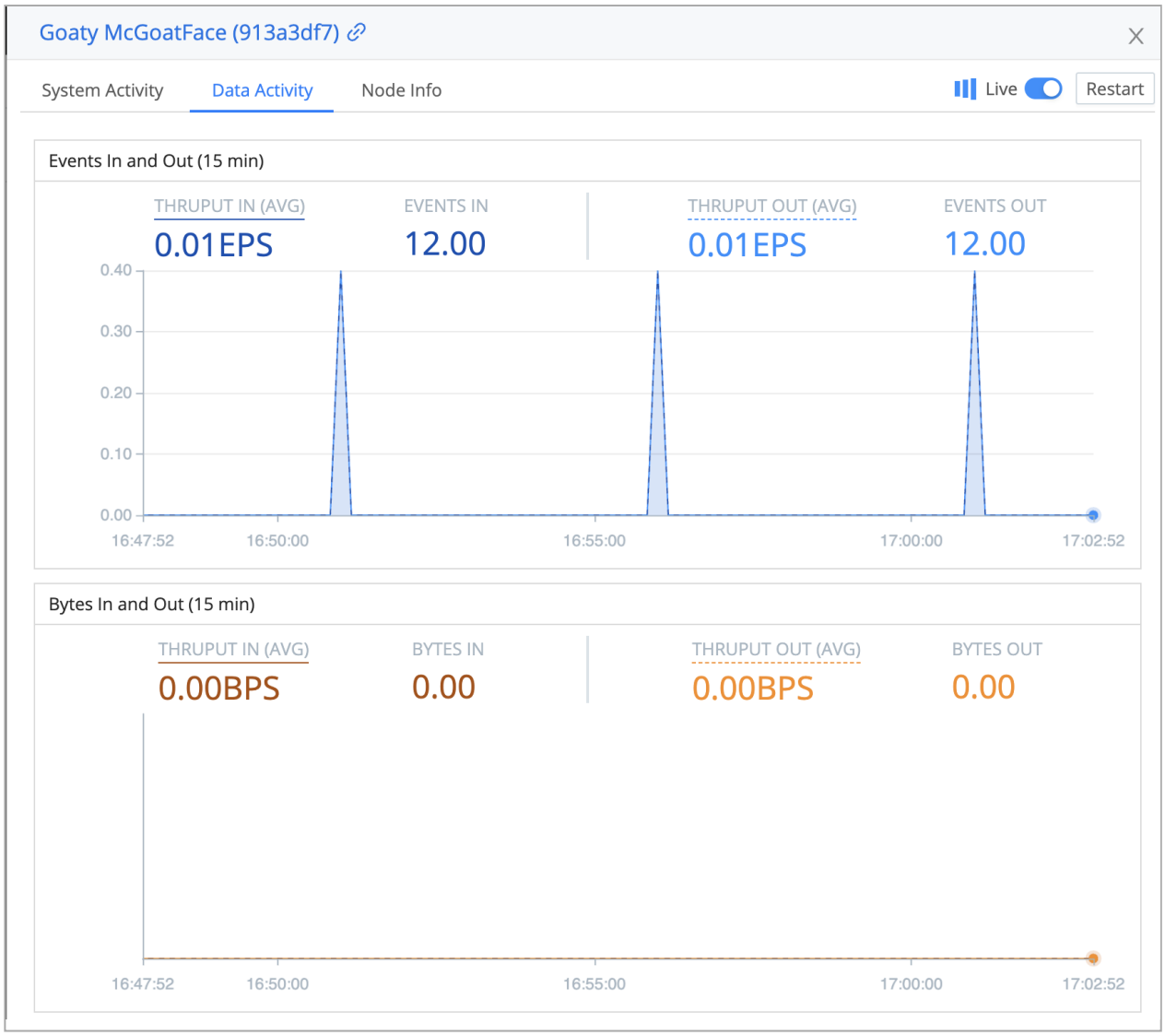
View Edge Node

Clicking any of the hexagons displays a modal providing details on the Edge Node, similar to teleporting into it, with the option to **Restart** the host. The **System Activity** tab displays details about the host's CPU, memory, network, and disk operations.



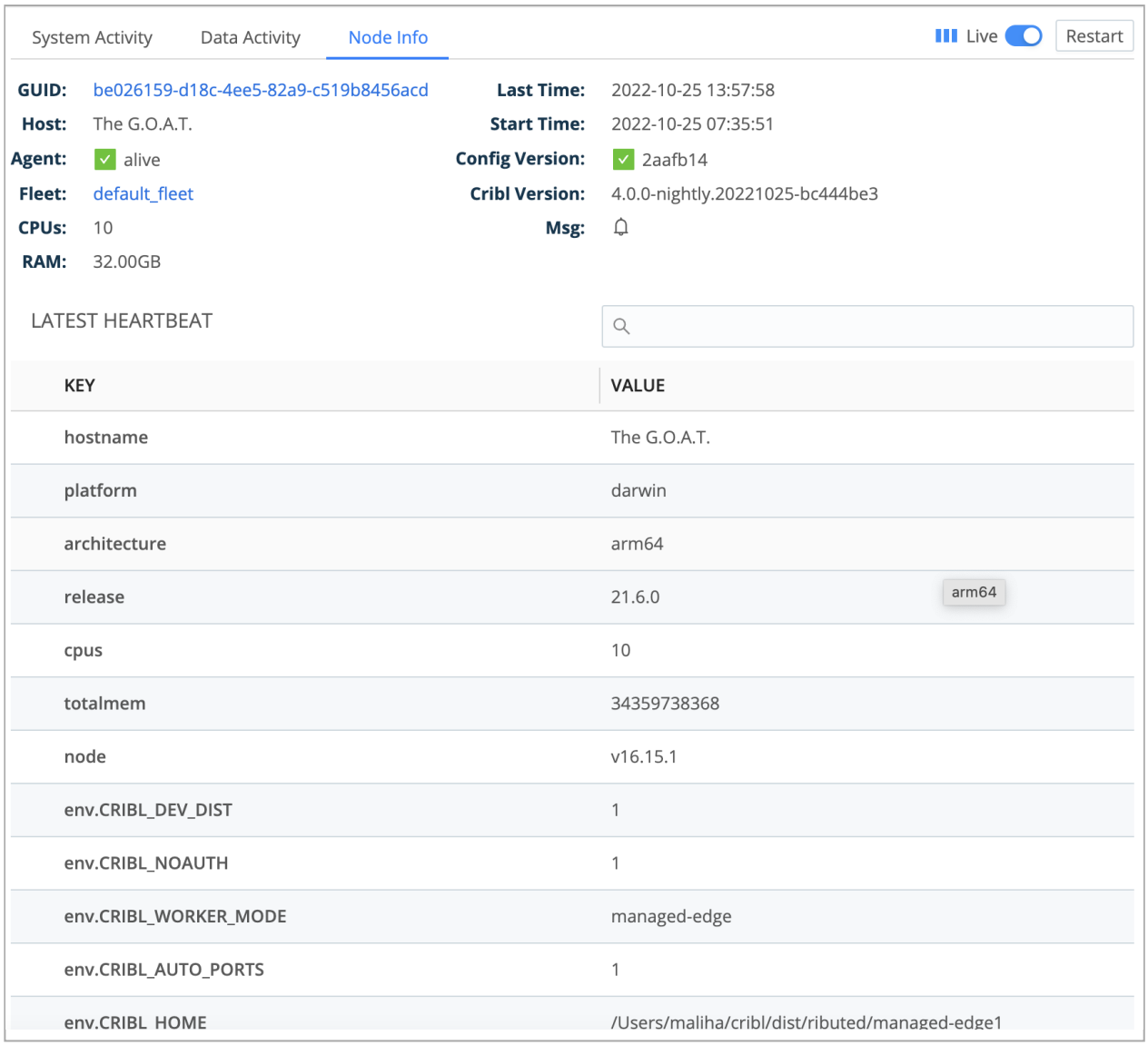
Edge Node System Activity

The **Data Activity** tab offers a view into the data flowing through the Edge Node.



Edge Node Data Activity

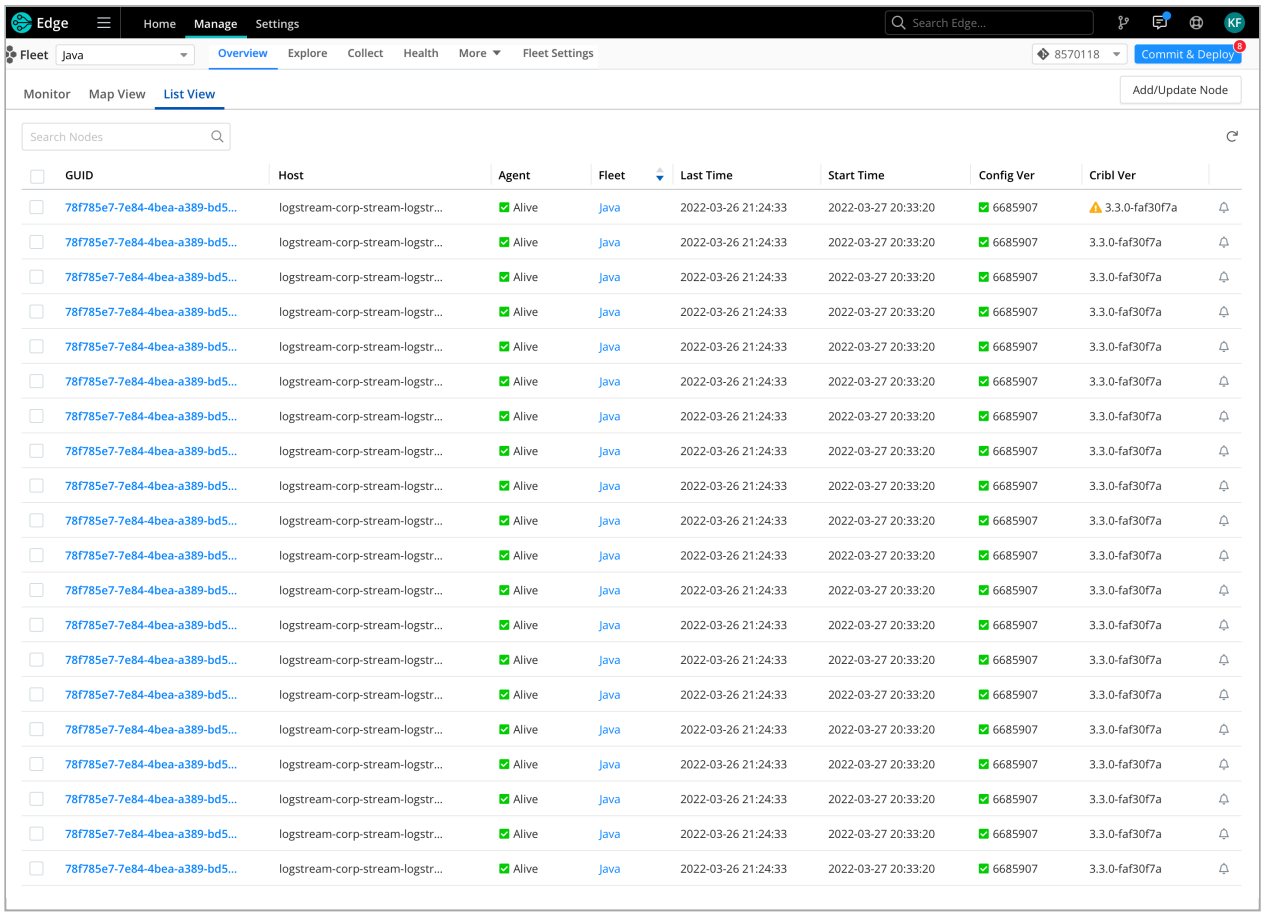
The **Node Info** tab offers Host/OS level information with a snapshot of the latest Heartbeat captured.



Edge Node info

Fleet List View

The **List View** tab displays a list of all the systems in the Fleet. This also serves as the "transporter room," allowing you to teleport into each of the Node's interfaces.



Edge Node list view

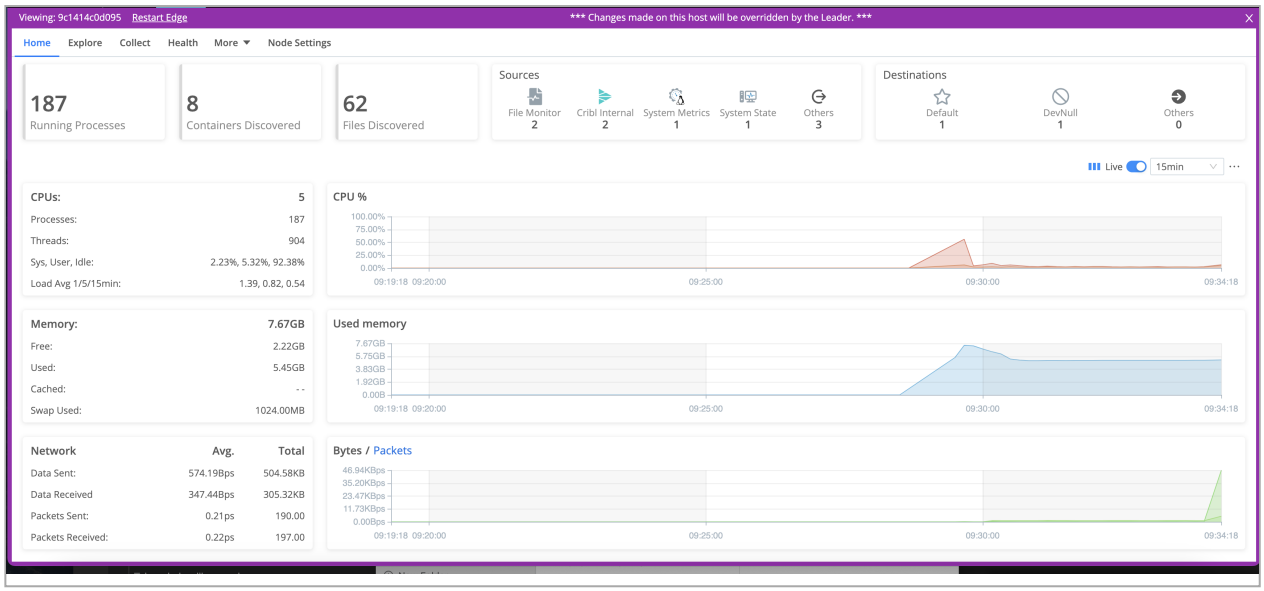
Click anywhere on a row to display a quick snapshot of the Edge Node's details, with an option to **Restart** the host.

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 autodiscovered, 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.

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 **Manage Edge Nodes** page, click the **X** close button on the upper right.

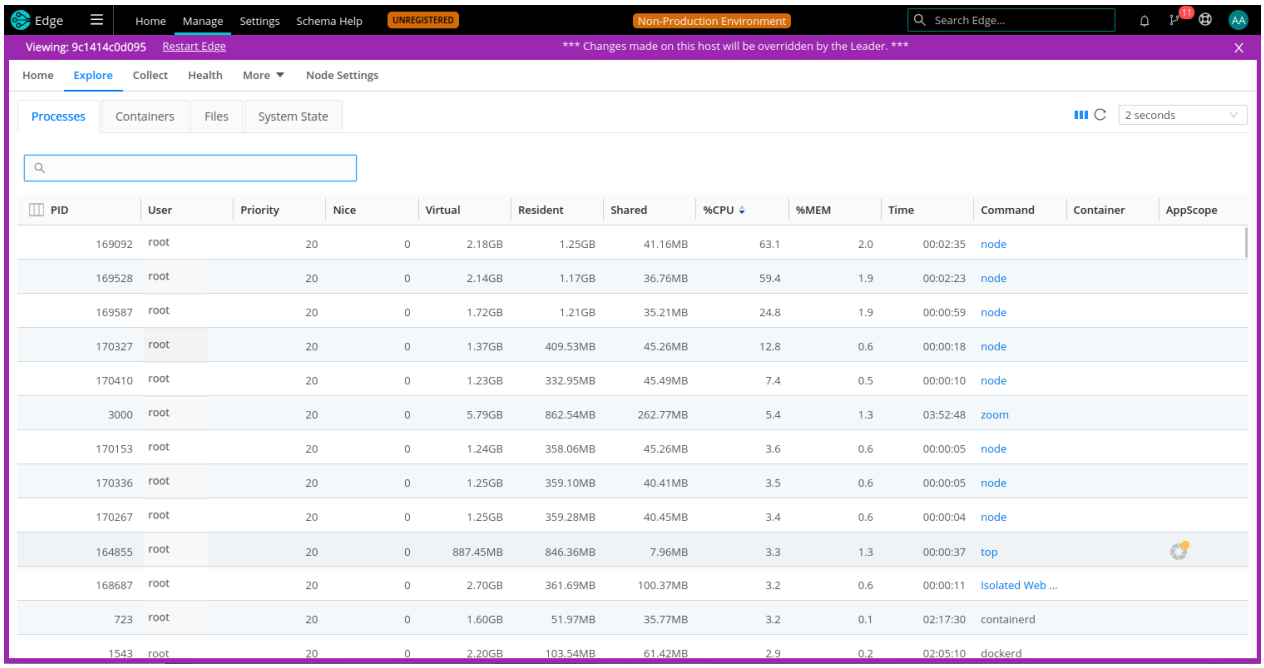


Teleporting into an Edge Node

> 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.

From the top nav, 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

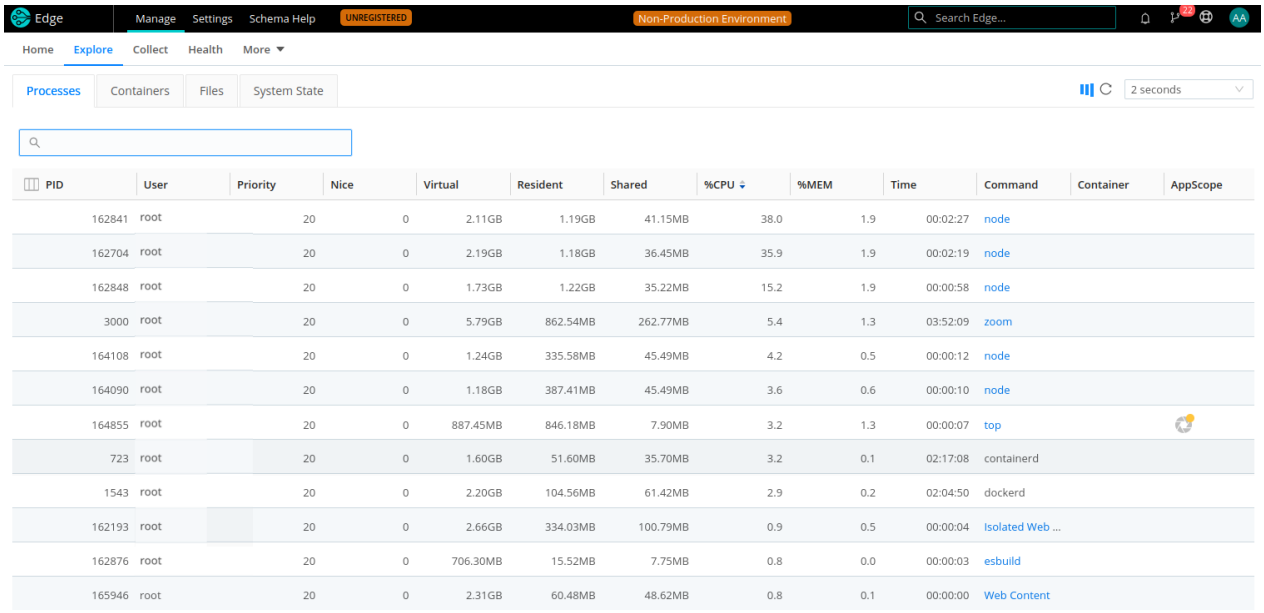


Explore an Edge Node

Let's explore each tab.

Processes

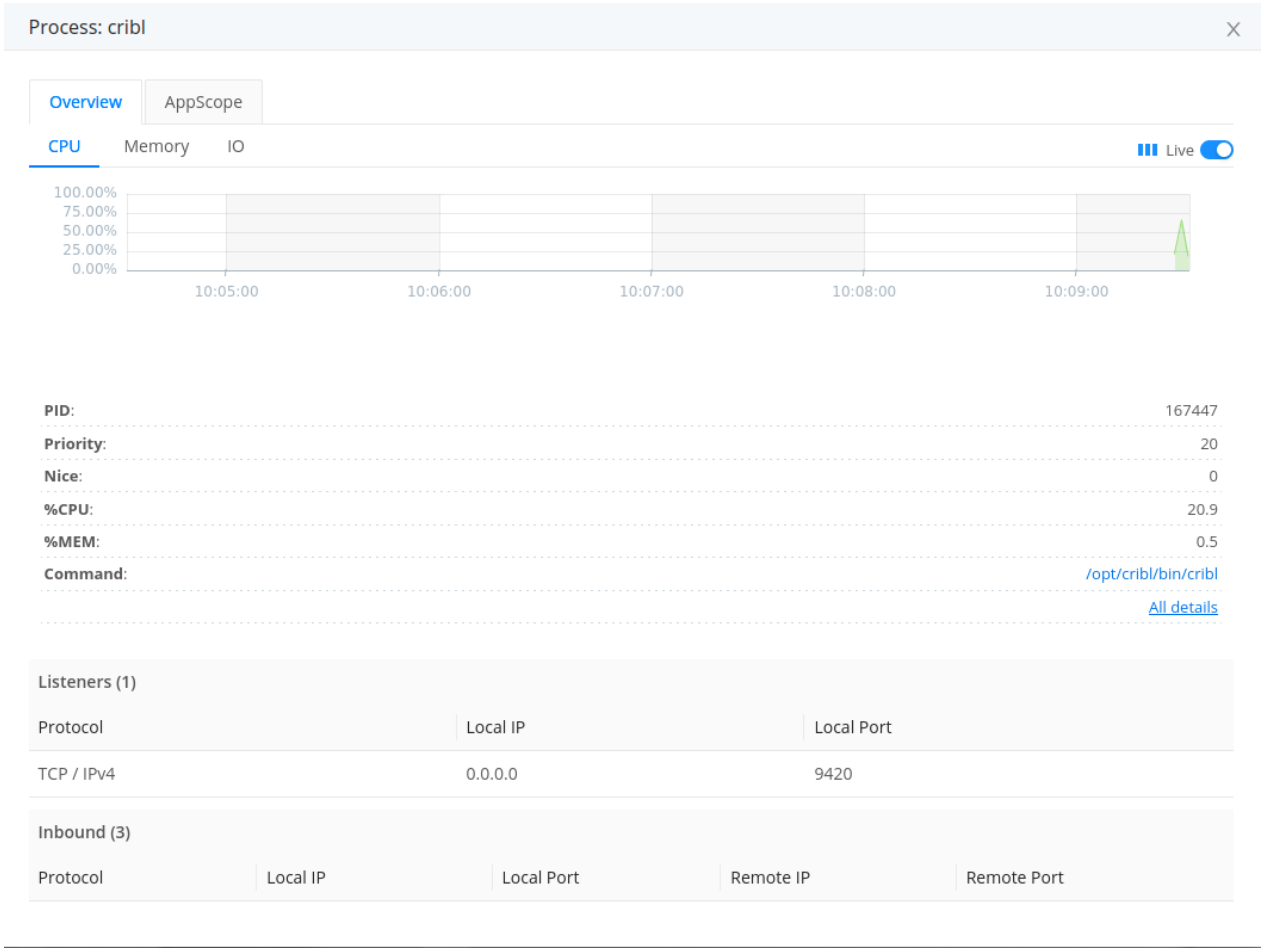
The **Processes** tab lists all the processes running on the Edge Node.



View Processes

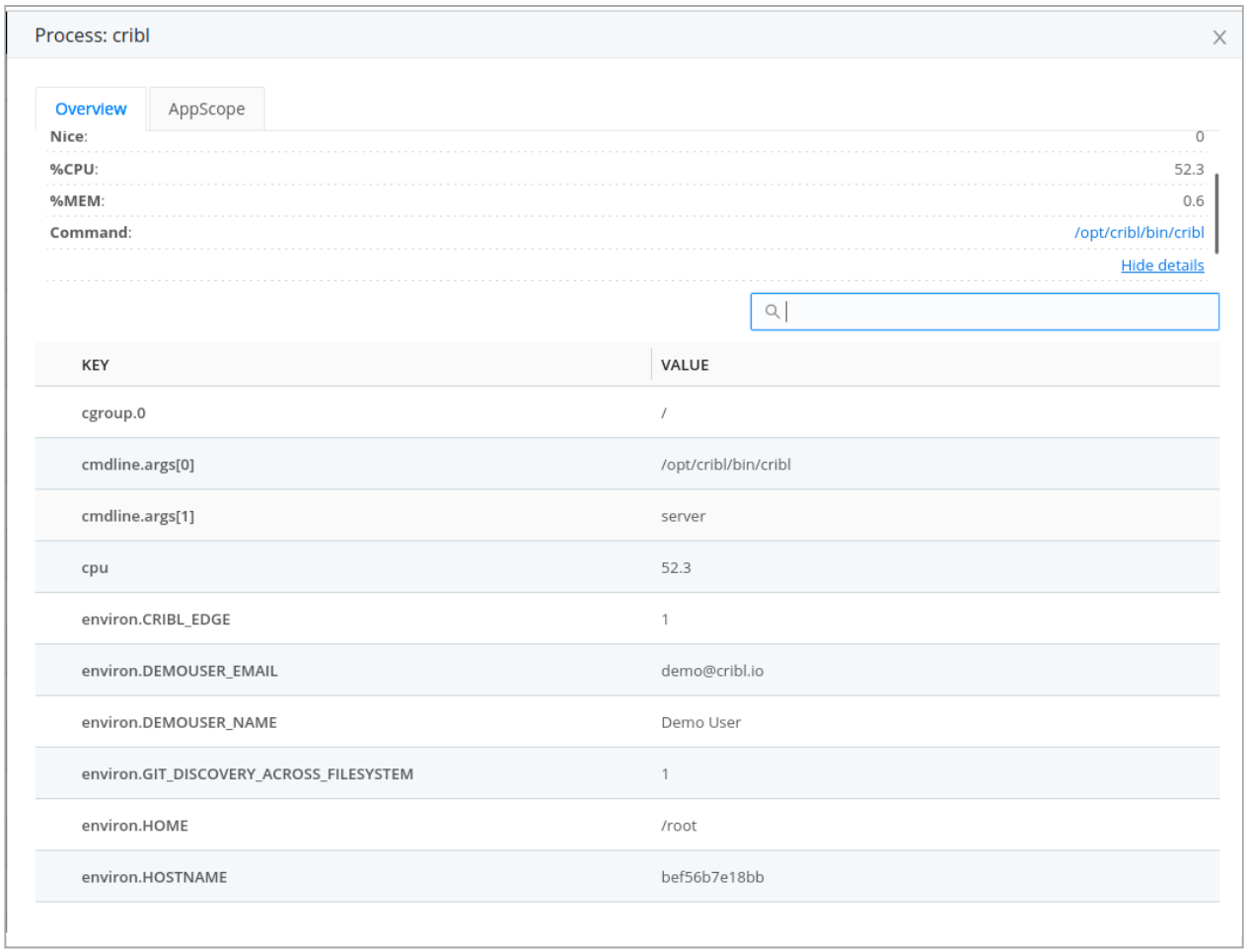
Click on any of the rows to open the **Process: <process_name>** drawer. In the drawer's default **Overview** tab, you'll find basic information on the process, including CPU, Memory, and IO graphs, along with tables for

active **Listening**, **Inbound**, and **Outbound** connections.



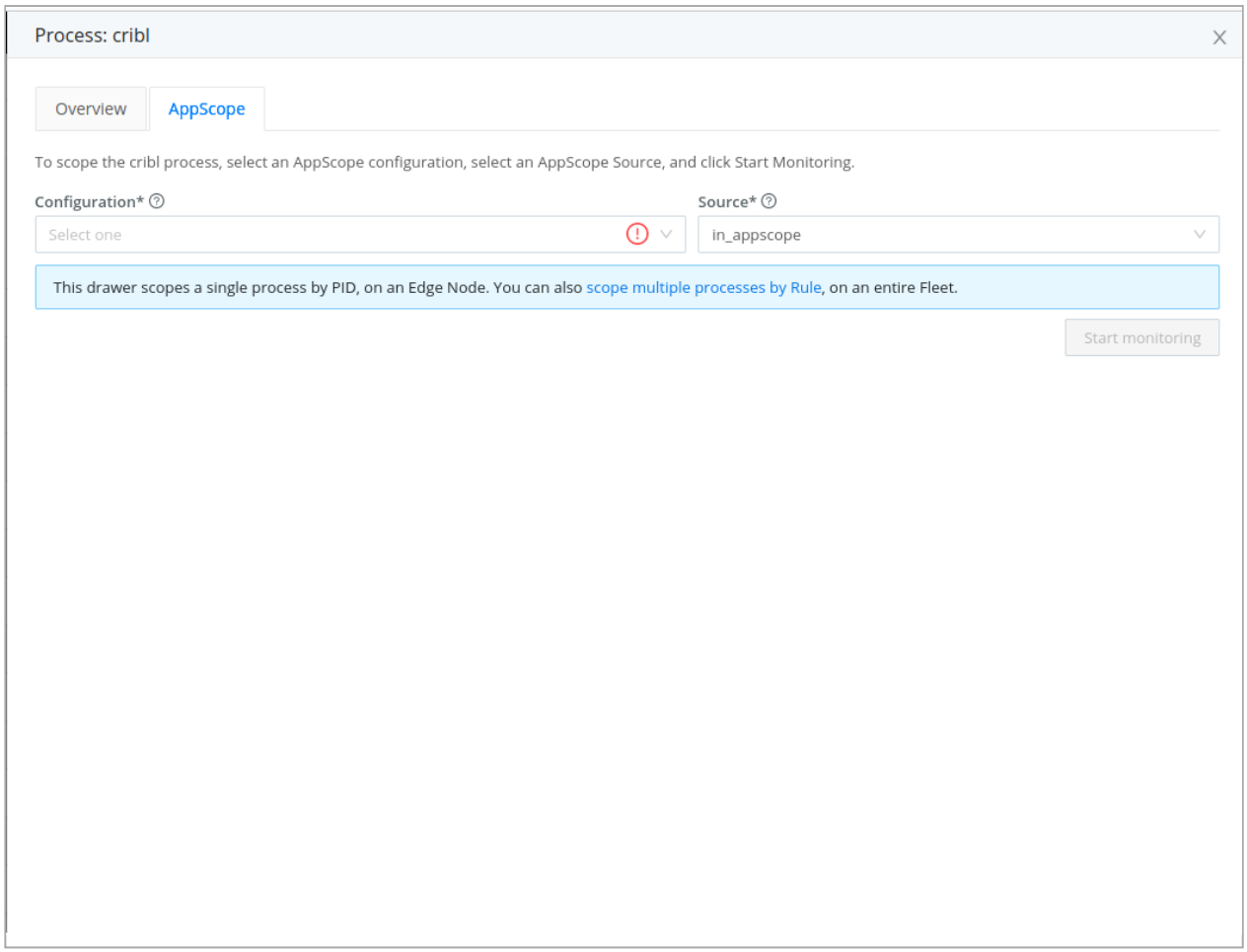
The Overview tab

In this tab, click **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

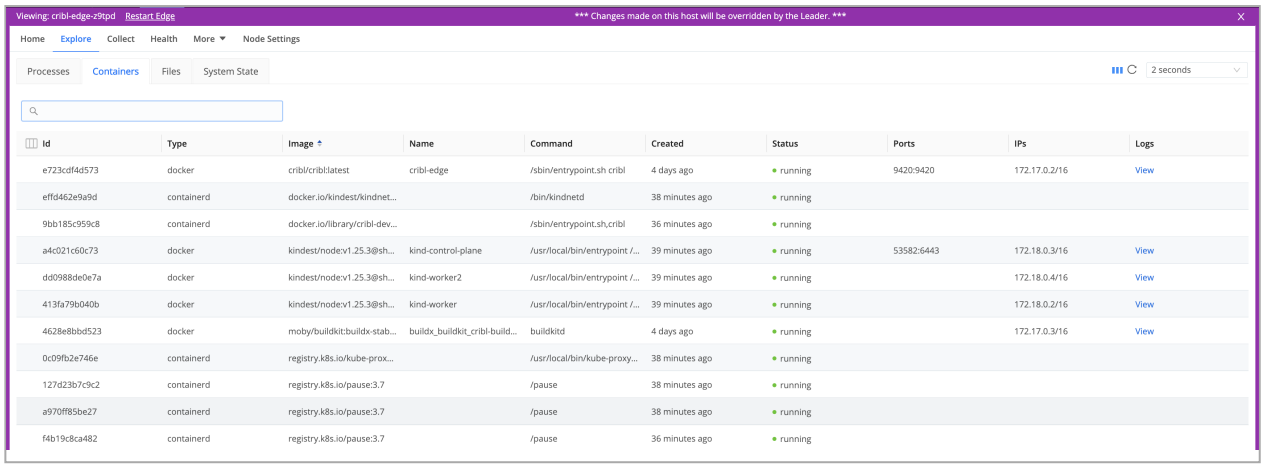
When a process is being scoped, back in the **Processes** tab you'll see that indicated in the process' entry in the **AppScope** column.

Containers

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

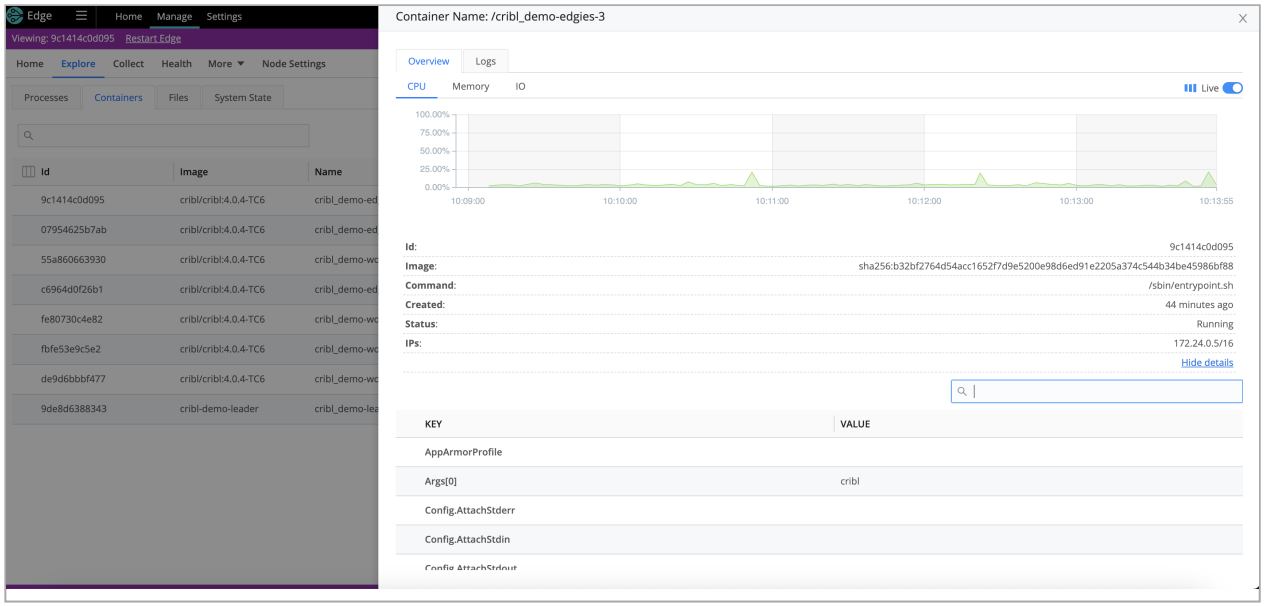
> Cribl Edge v.4.1.x and later supports both Docker and `containerd` runtimes.

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



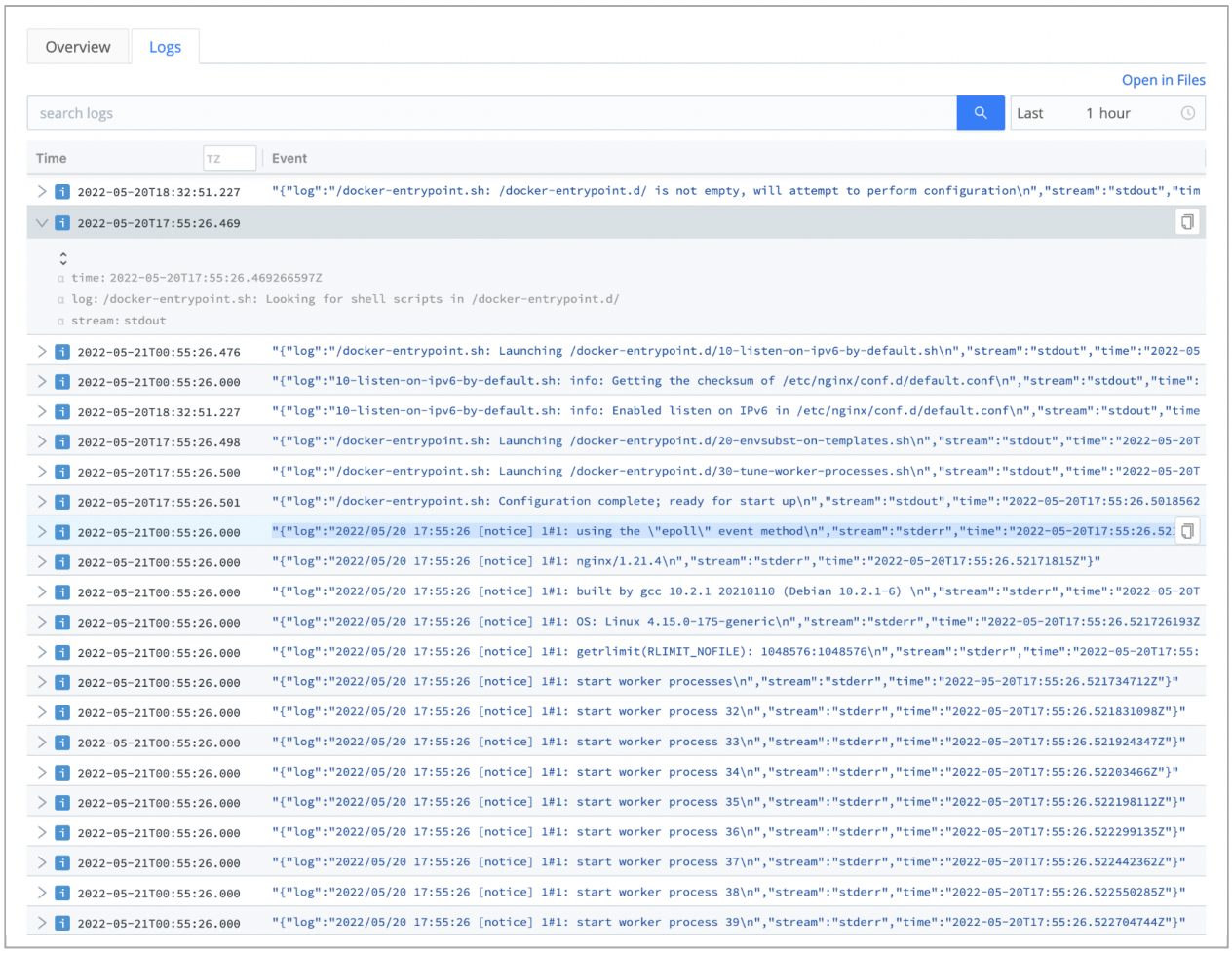
Containers overview

Click any container to view more details:



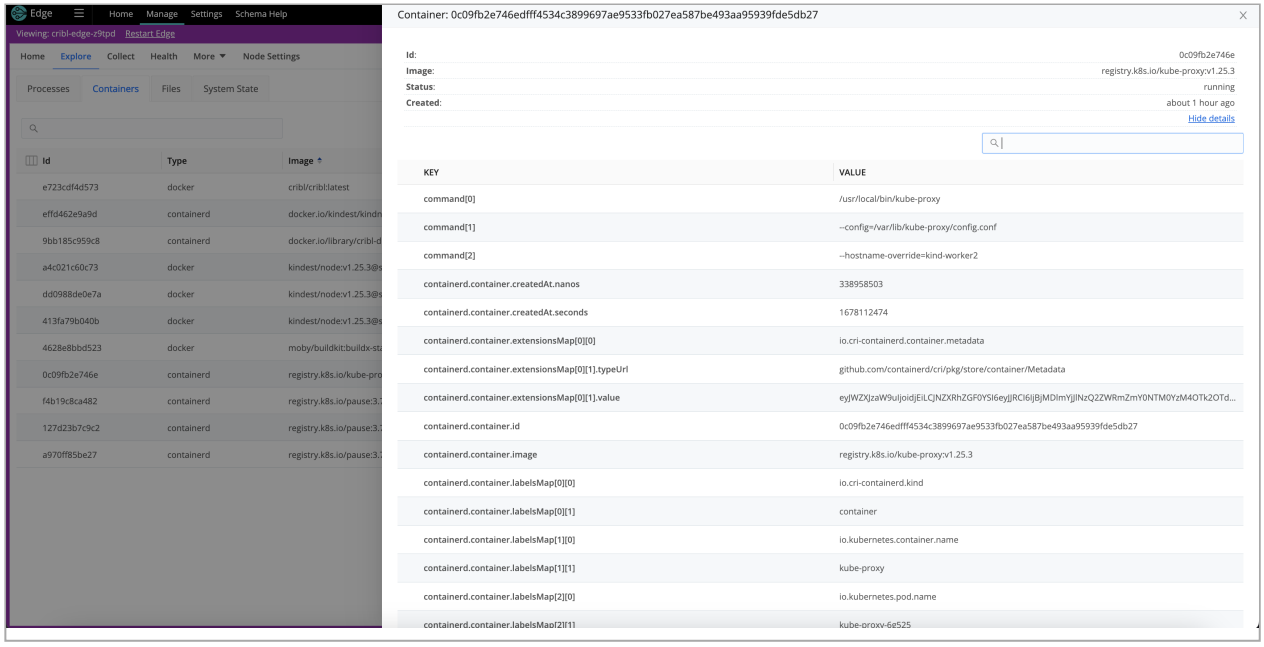
Container details

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



View Docker Container logs

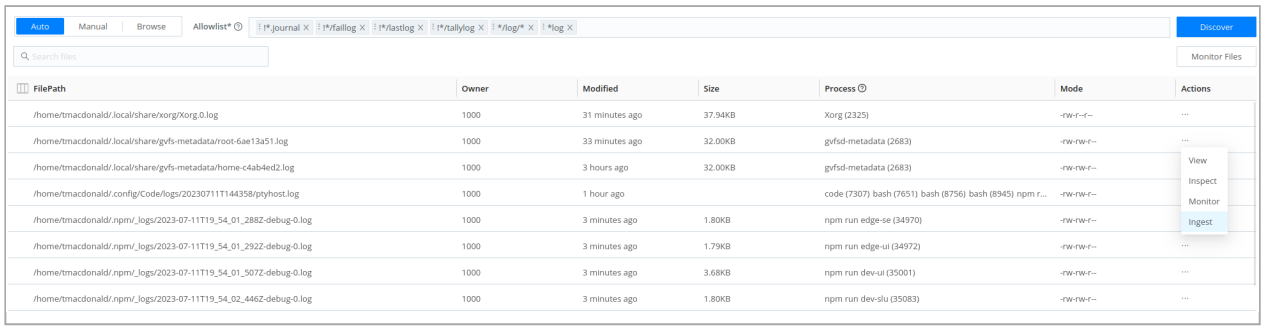
The screenshot below shows `containerd` details, which don't include charts or logs.



View containerd logs

Files

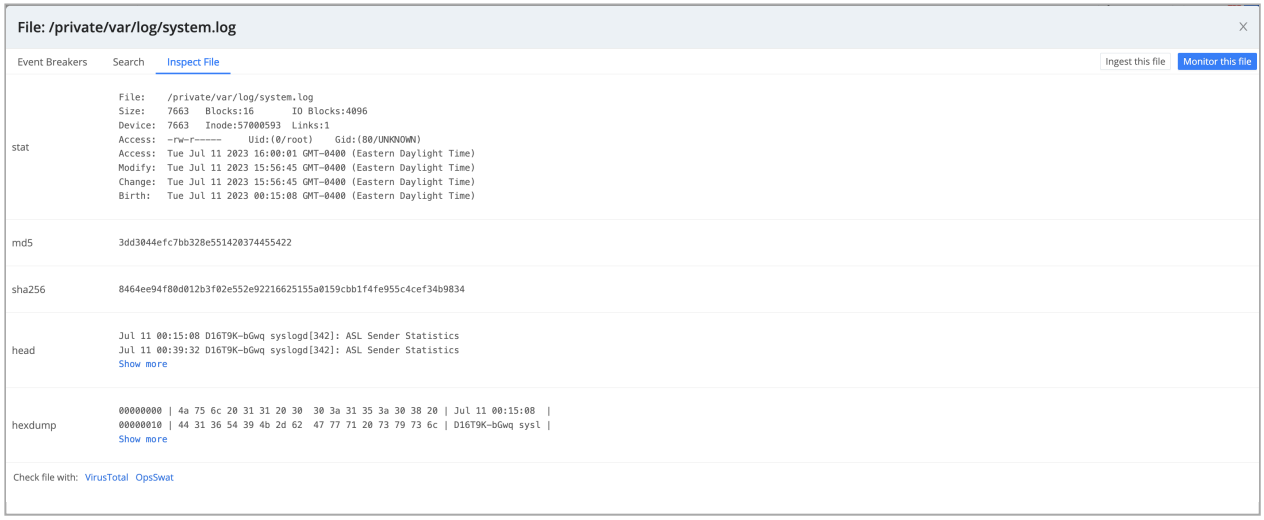
The **Files** tab lists all the log files being actively written to by running applications that Cribl Edge has auto-discovered. You can also specify a list of directories and files to actively monitor.



Explore files

The **Actions** column allows you to:

- **View**: Displays a representation of the lines this column contains. You can also 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. If the file appears suspicious, click **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

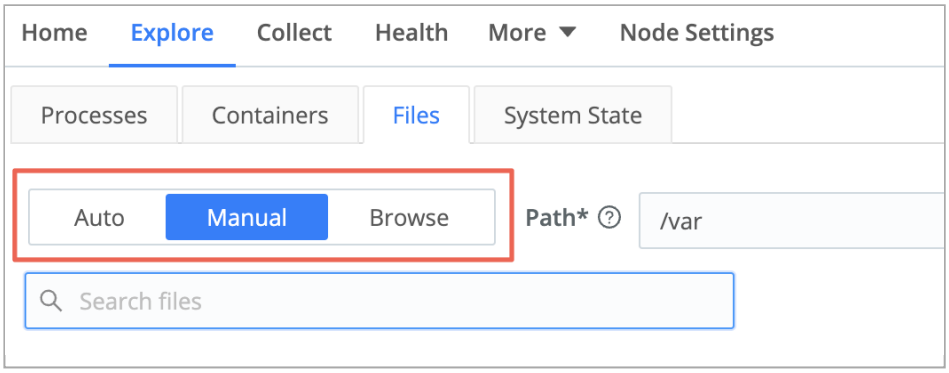
configurations.

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

File Discovery Modes

Click 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

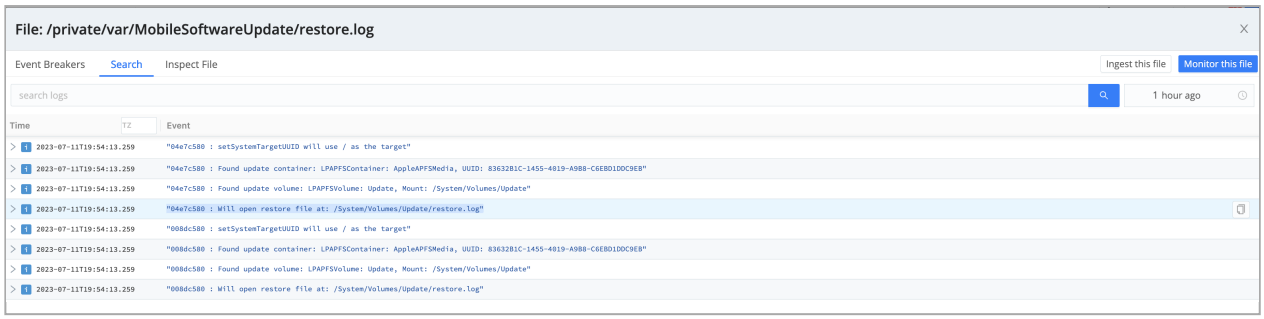
Path

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

Allowlist

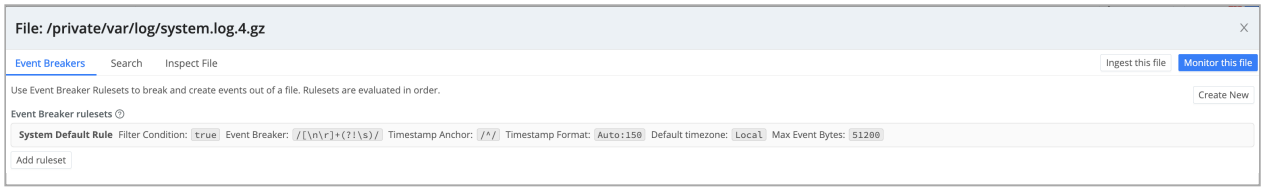
The **Allowlist** field, available with Auto and Manual discovery, supports wildcard syntax, and supports the exclamation mark (`!`) for negation. For example, you can use `!*cribl*access.log` to prevent Cribl Edge from discovering its own access log. The default filters are `*/log/*` and `*log`.

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 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.

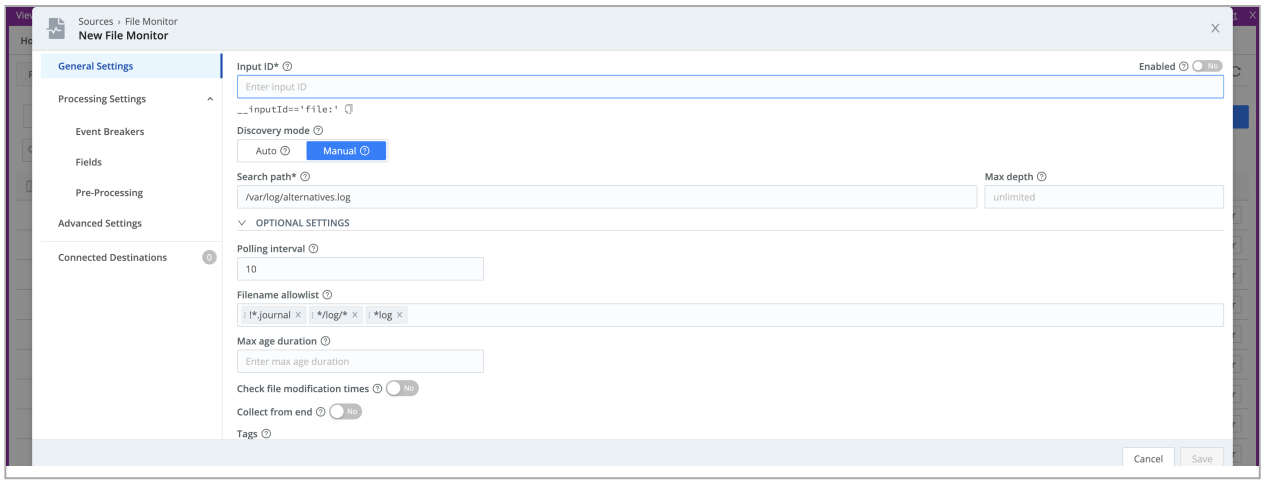
Max Depth

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.

Monitoring a File

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

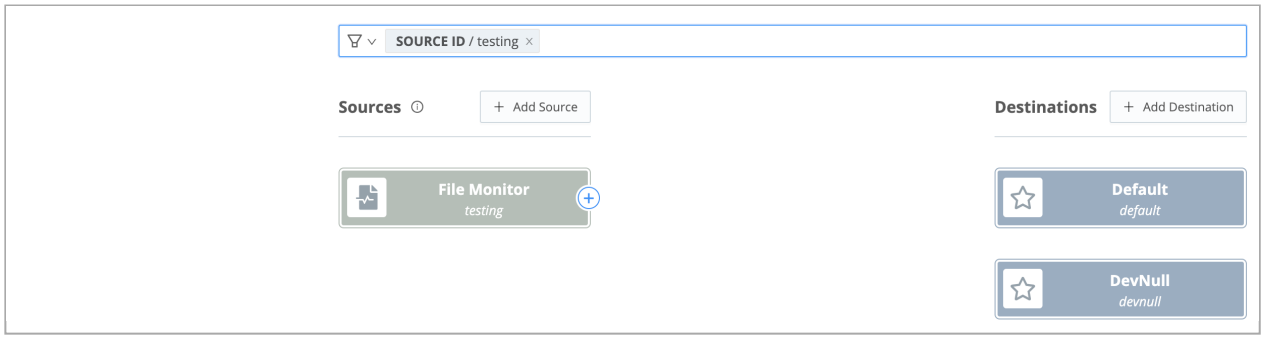


New File Monitor modal

The **Monitor** feature automatically prepopulates 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

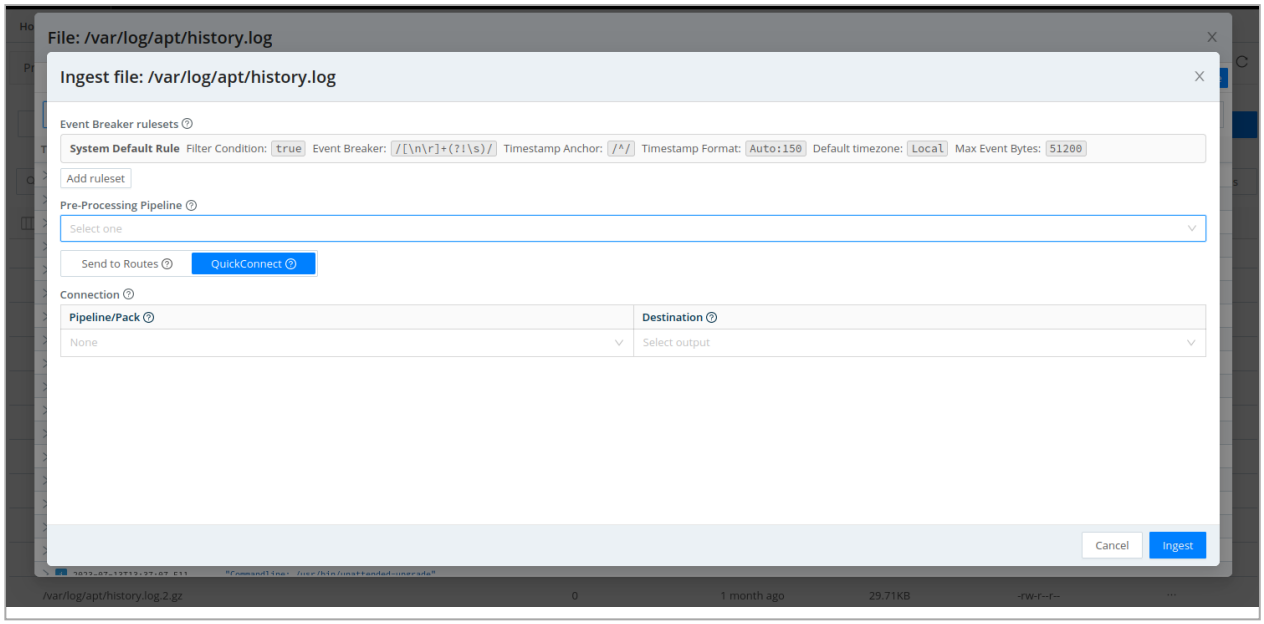
For further details, see File Monitor and QuickConnect.

Ingesting a File

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

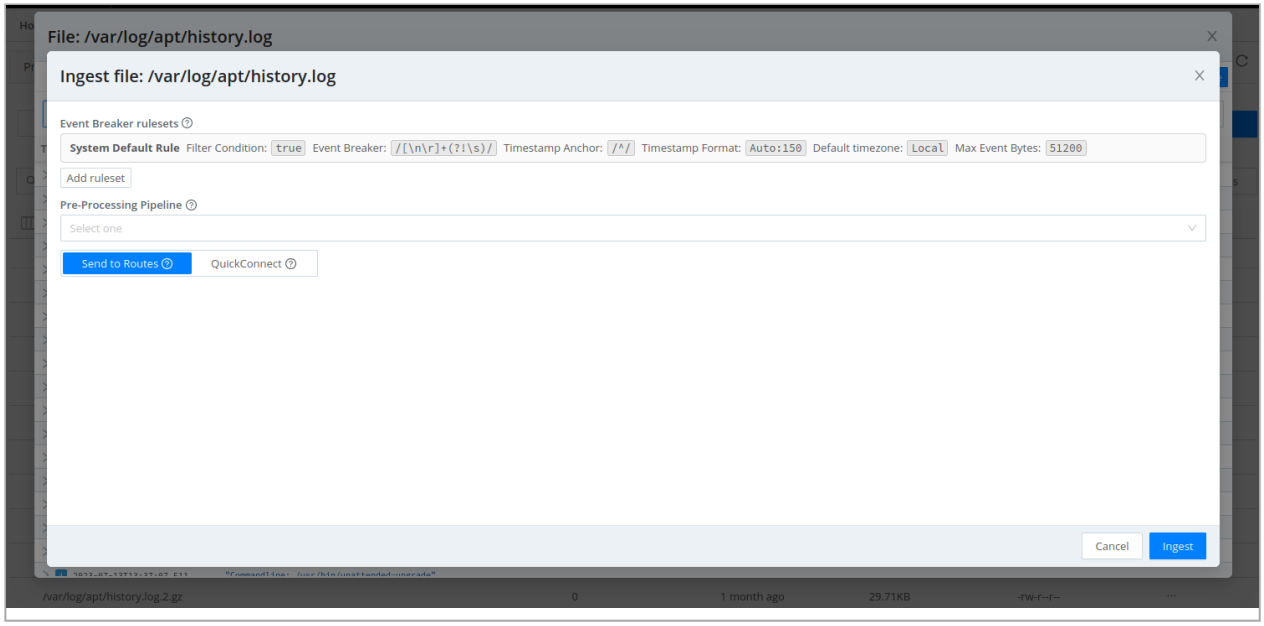
You have two options:

- 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? 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, click **Add ruleset**, then select the desired ruleset from the **Event Breaker rulesets** drop-down.

- To create a new ruleset, click **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

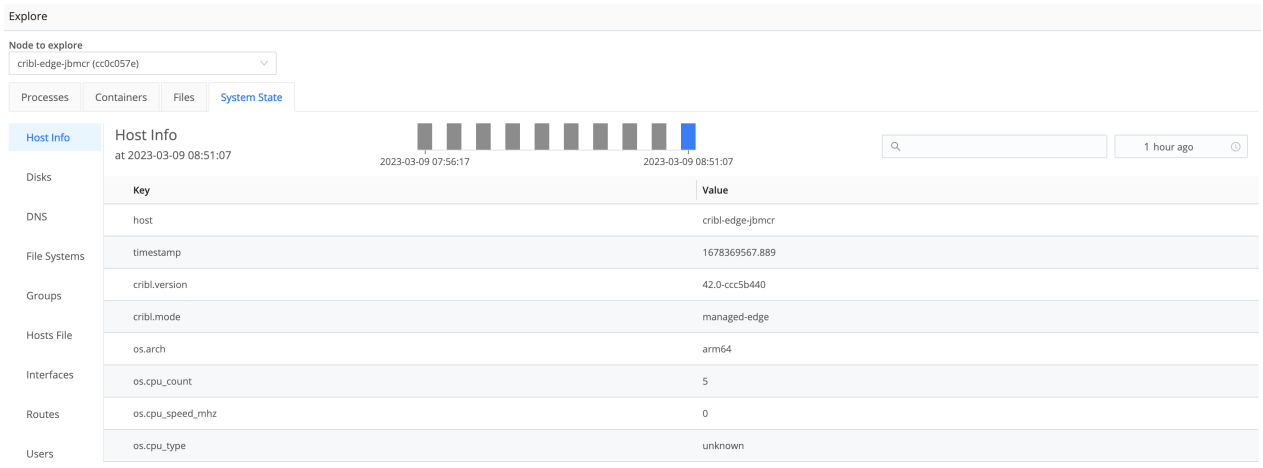
The **System State** upper tab provides access to these left tabs:

- 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**.



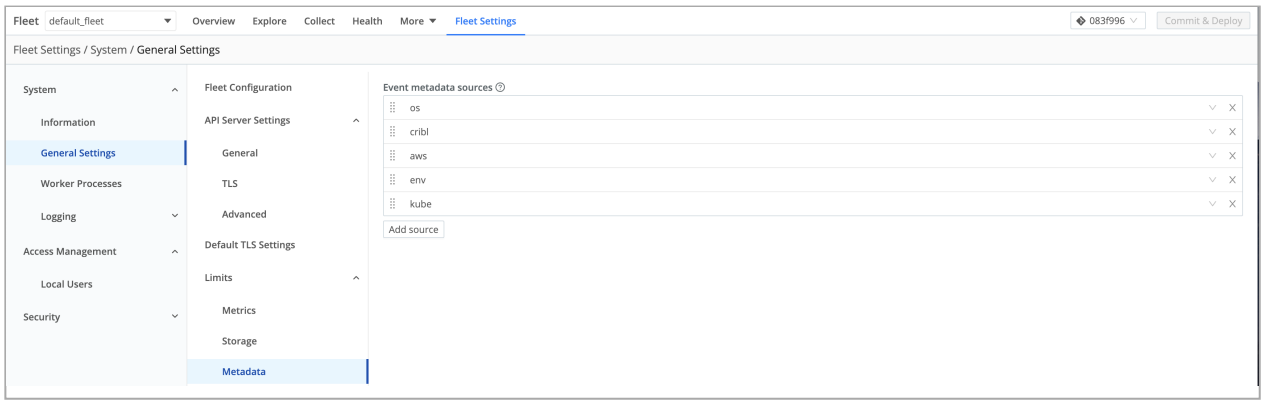
Host info/metadata collected

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:

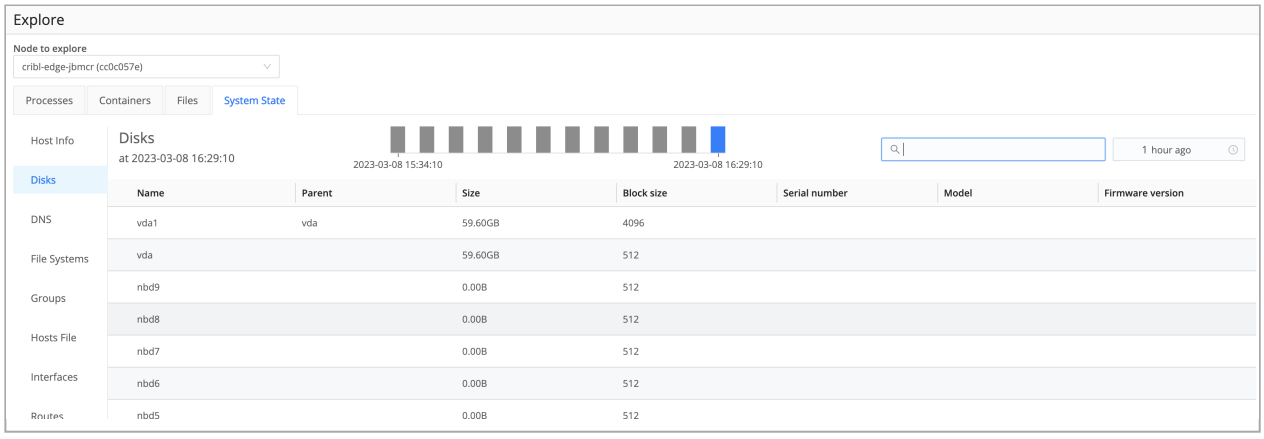
- `os` : Reports details for the host OS and host machine, like OS version, kernel version, CPU and memory resources, hostname, network addresses, etc.
- `cribl` : Reports the Cribl Edge version, mode, Fleet for managed instances, and config version.
- `aws` : Reports details for an EC2 instance, including 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

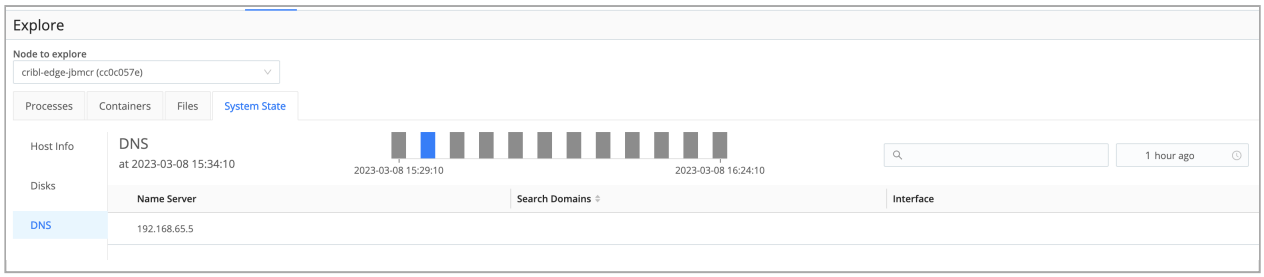
The **Disks** tab displays the inventory of physical disks and their partitions on the host system.



Disks tab

DNS

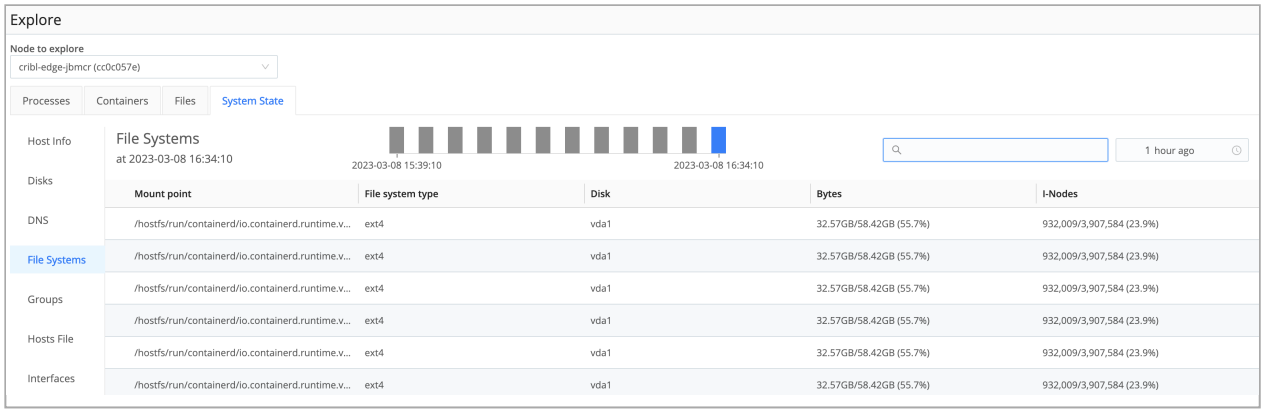
The **DNS** tab lists the host system's DNS resolvers and search entries.



DNS tab

File Systems

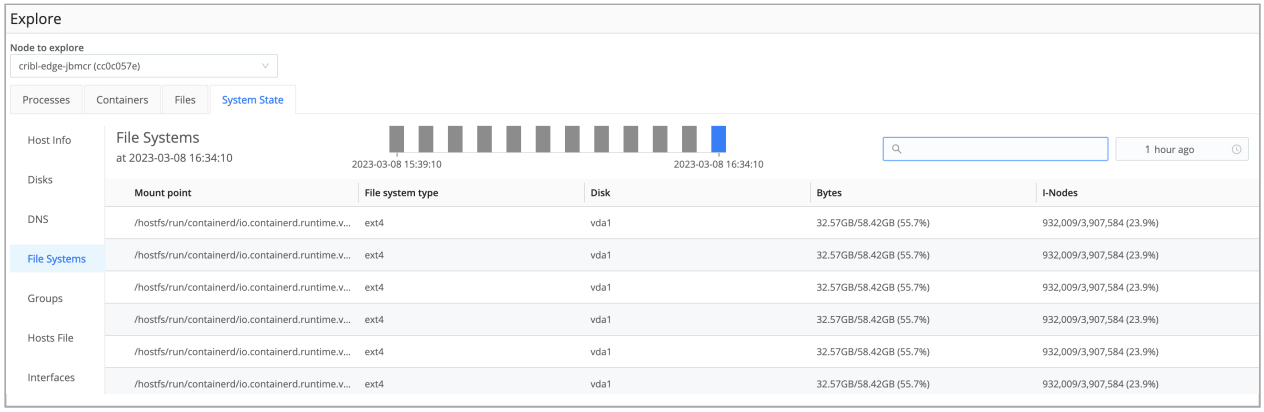
The **File Systems** tab displays an inventory of the mounted file systems on the host system.



File Systems tab

Firewall

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



Firewall tab

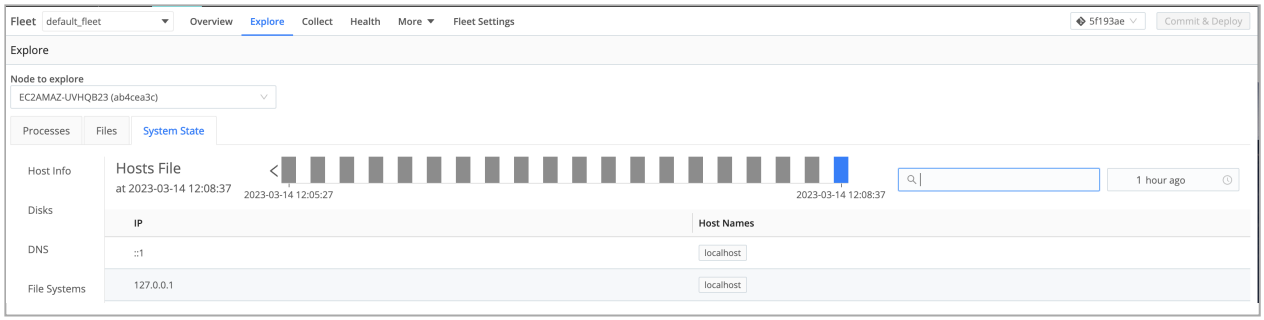
Groups

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

Groups tab

Hosts File

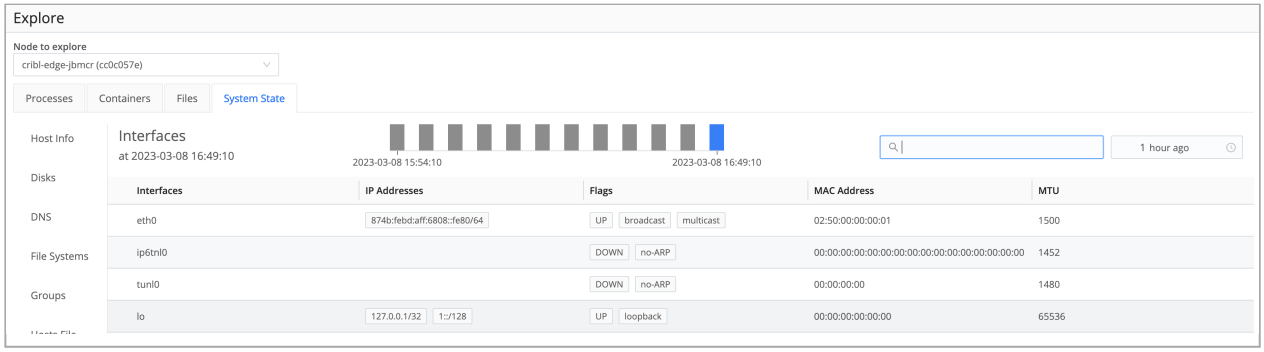
The **Hosts File** tab displays the host system's current state.



Hosts File tab

Interfaces

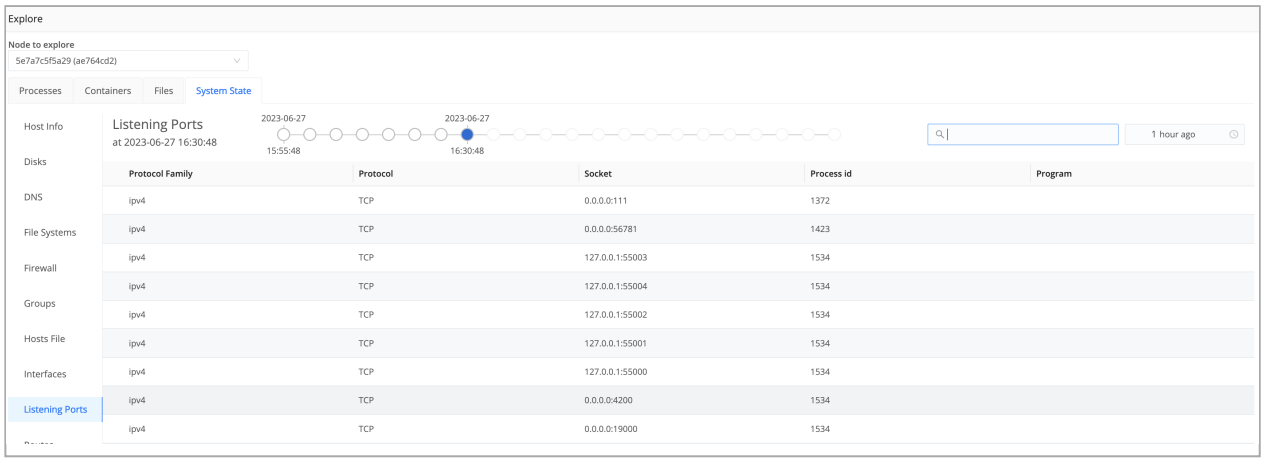
The **Interfaces** tab displays a list of each of the network interfaces on the host system.



Interfaces tab

Listening Ports

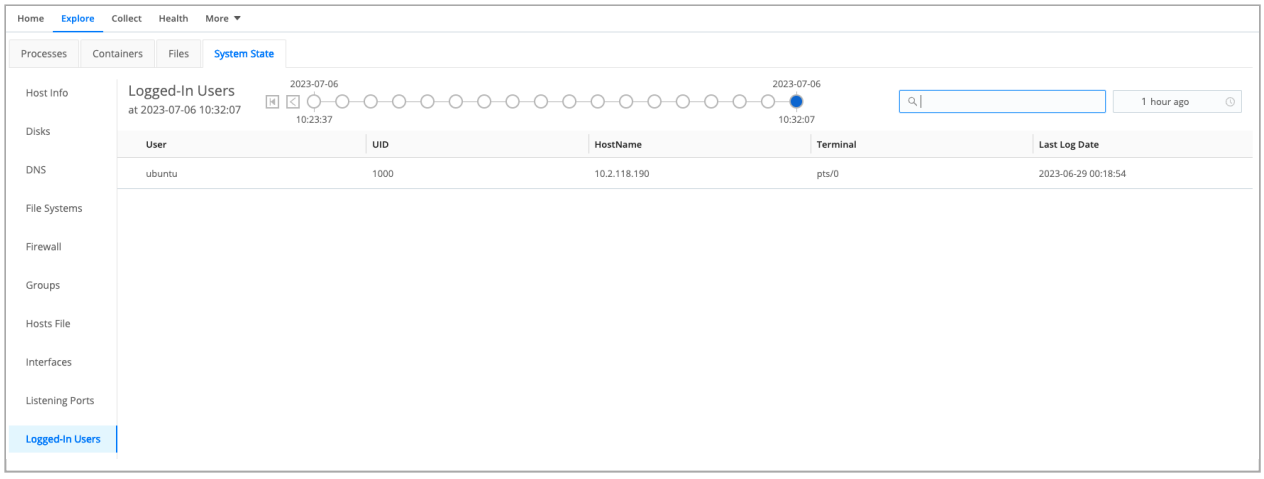
The **Listening Ports** tab displays a list of listening ports and their associated process identifier (pid).



Listening Ports tab

Logged-In Users

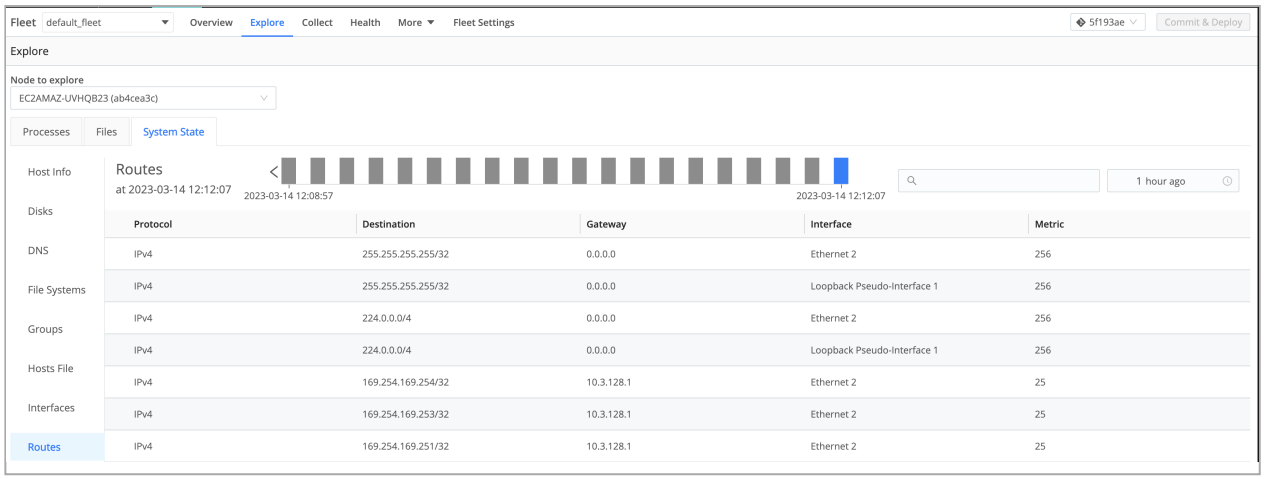
The **Logged-In Users** tab displays a list of currently logged-in users on the host.



Logged-In Users tab

Routes

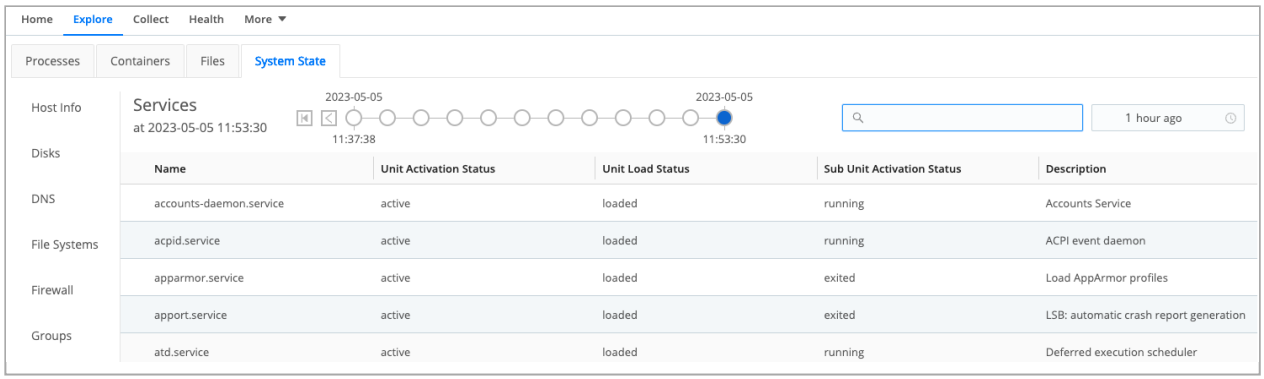
The **Routes** tab displays entries from the network routes on the host system.



Hosts Routes tab

Services

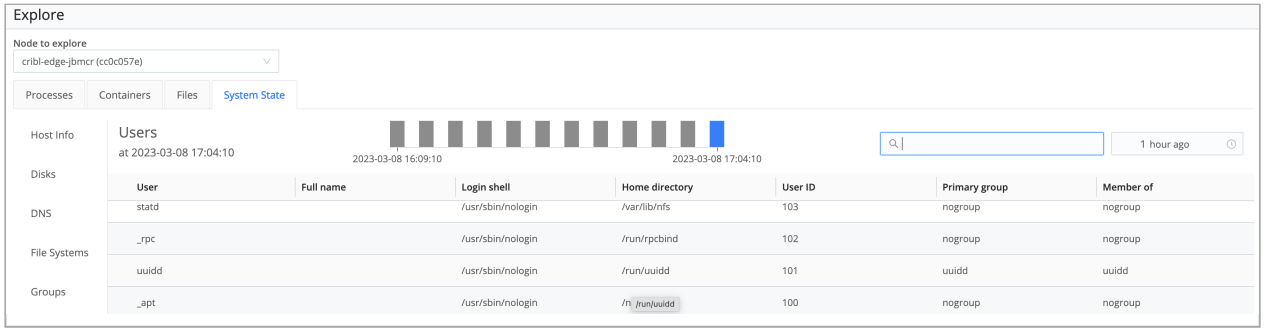
The **Services** tab displays a list of each configured service (e.g. `systemd` and `initd`) along with their running status.



Services tab

Users

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



Cribl University offers a course titled Collecting Data with Linux that provides a good overview of working with Linux. To follow the direct course link, first log into your Cribl University account. (To create an account, click 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.

`__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.

1.4. Exploring 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 Windows Server 2016, 2019, or 2022, to collect events via the Windows Events API.

Limitations

Cribl Edge on Windows is currently subject to the following limitations. We will update this page as we enable more features.

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** (top nav) > **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

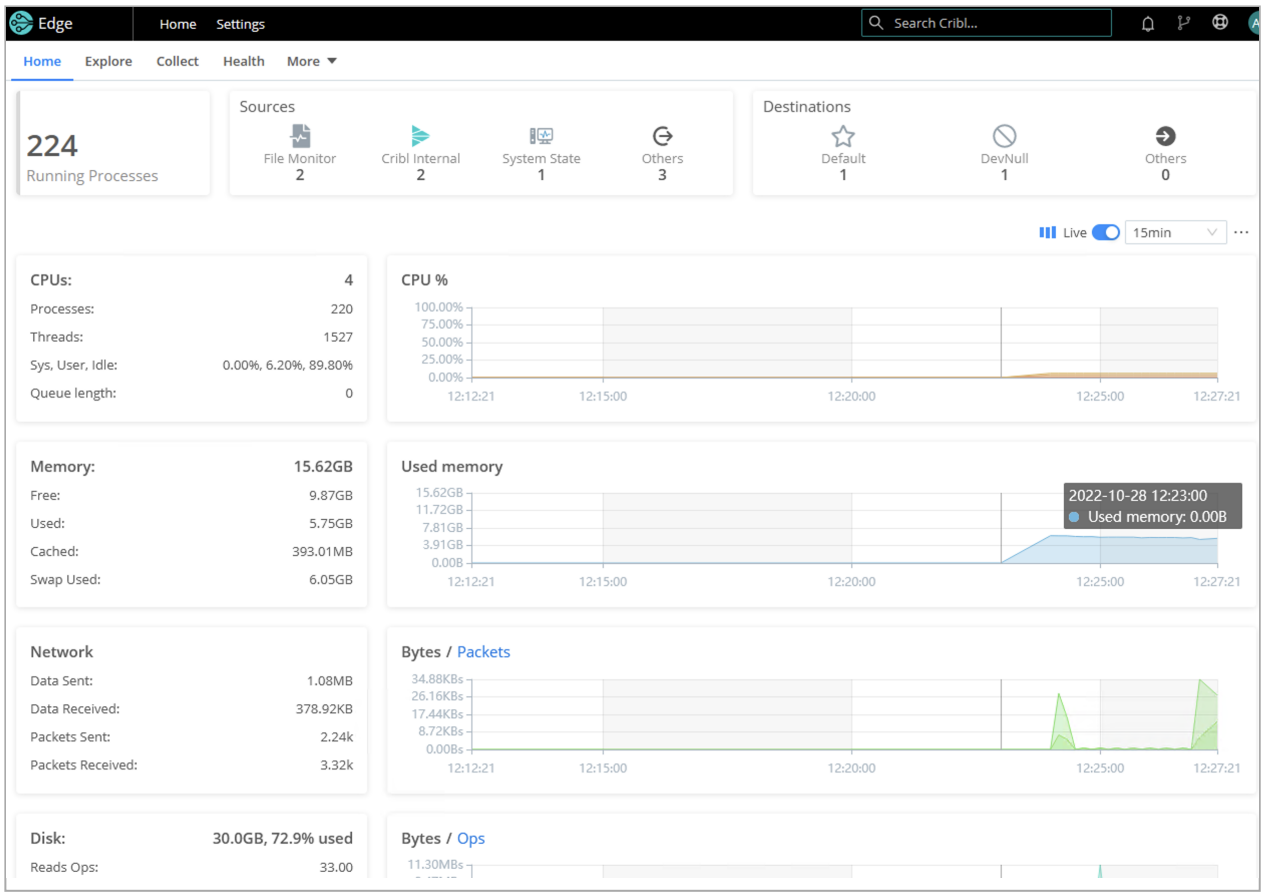
The Grok Function is unavailable on Windows. If you include it in a Pipeline, Cribl Edge processing will skip over it.

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.

Accessing Cribl Edge on Windows

When you first log into 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 can manually discover and explore other data of interest.

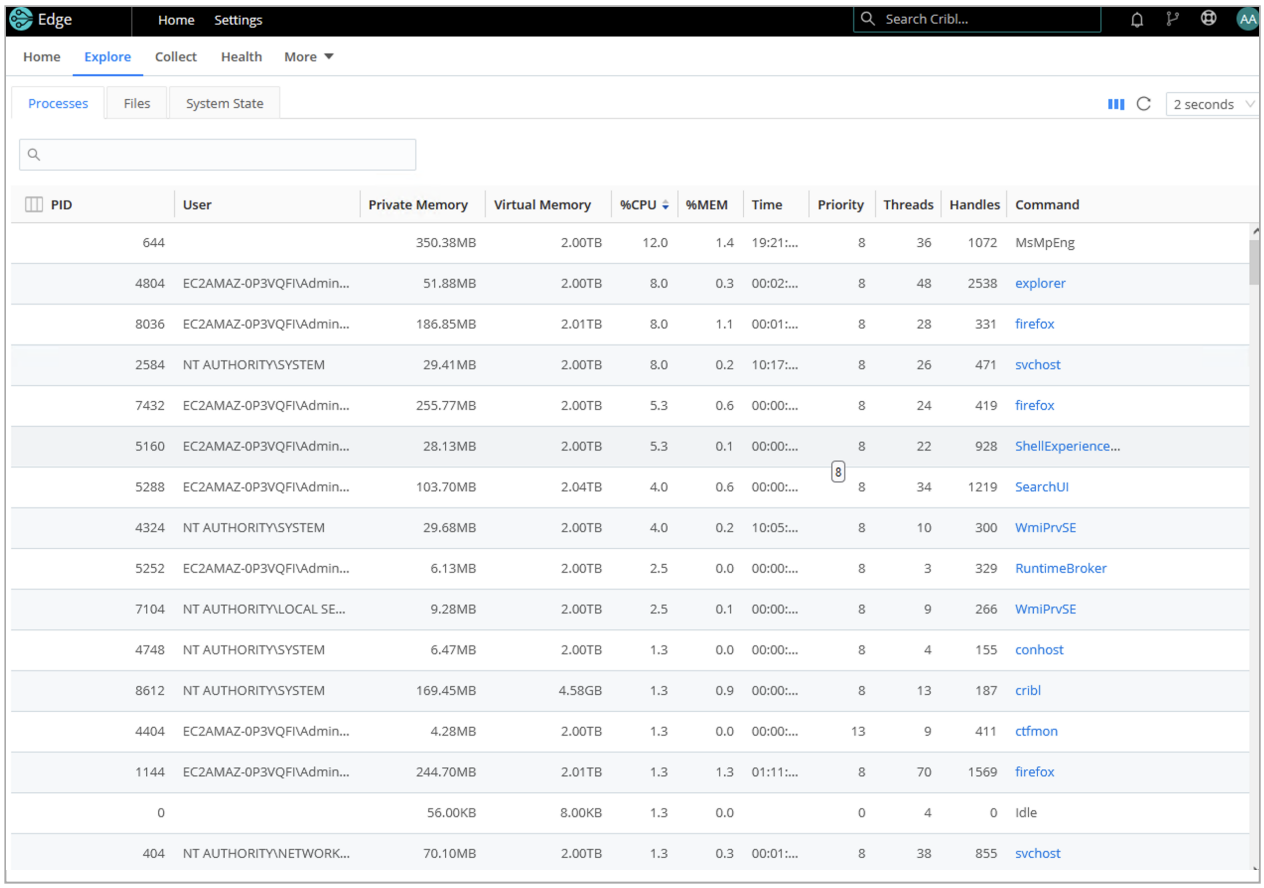


From the **Explore** page, you can view more details on your node via the following tabs:

- Processes
- Files
- Host Info/Metadata

Processes

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



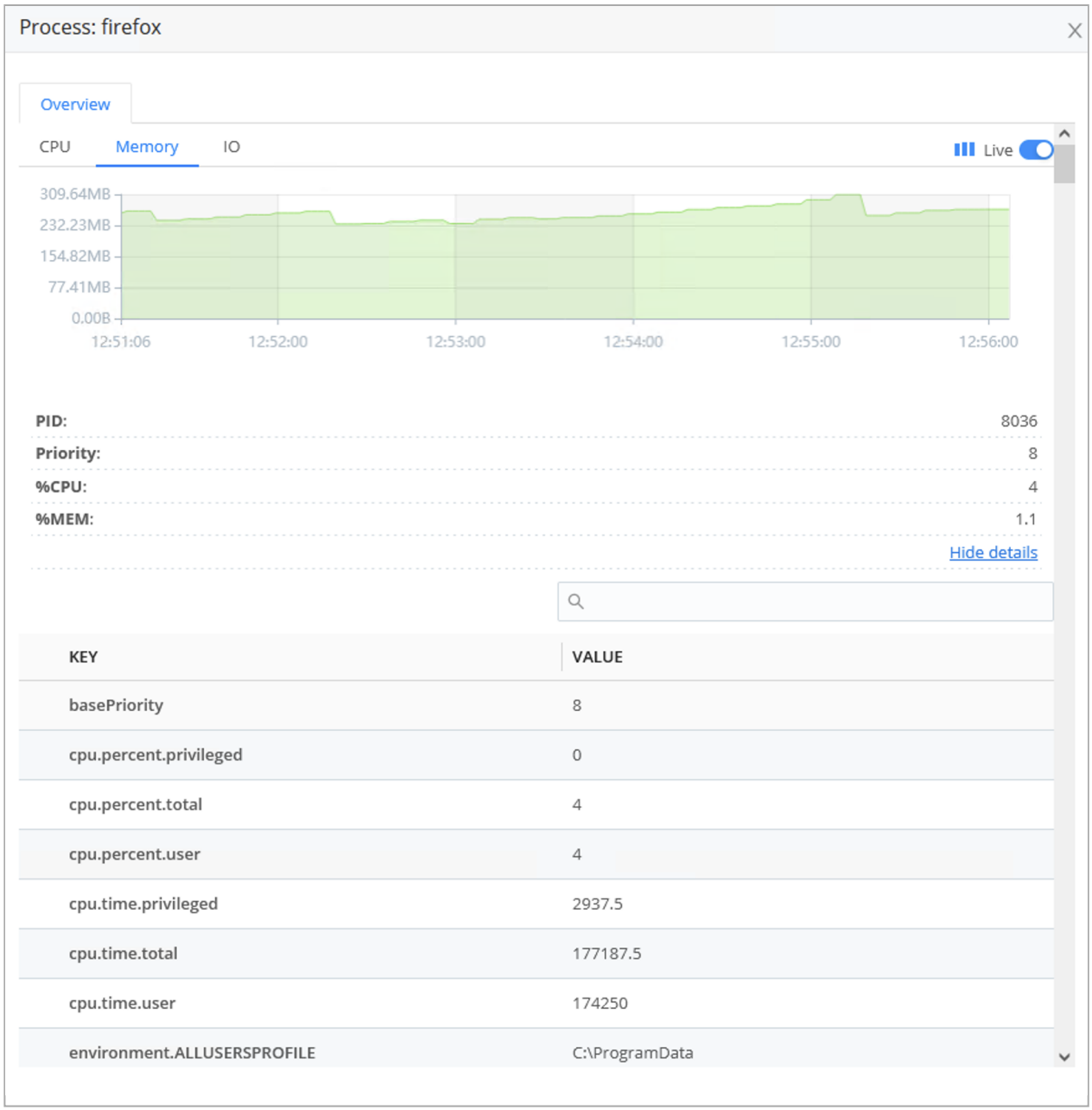
PID	User	Private Memory	Virtual Memory	%CPU	%MEM	Time	Priority	Threads	Handles	Command
644		350.38MB	2.00TB	12.0	1.4	19:21:...	8	36	1072	MsMpEng
4804	EC2AMAZ-0P3VQFI\Admin...	51.88MB	2.00TB	8.0	0.3	00:02:...	8	48	2538	explorer
8036	EC2AMAZ-0P3VQFI\Admin...	186.85MB	2.01TB	8.0	1.1	00:01:...	8	28	331	firefox
2584	NT AUTHORITY\SYSTEM	29.41MB	2.00TB	8.0	0.2	10:17:...	8	26	471	svchost
7432	EC2AMAZ-0P3VQFI\Admin...	255.77MB	2.00TB	5.3	0.6	00:00:...	8	24	419	firefox
5160	EC2AMAZ-0P3VQFI\Admin...	28.13MB	2.00TB	5.3	0.1	00:00:...	8	22	928	ShellExperience...
5288	EC2AMAZ-0P3VQFI\Admin...	103.70MB	2.04TB	4.0	0.6	00:00:...	8	34	1219	SearchUI
4324	NT AUTHORITY\SYSTEM	29.68MB	2.00TB	4.0	0.2	10:05:...	8	10	300	WmiPrvSE
5252	EC2AMAZ-0P3VQFI\Admin...	6.13MB	2.00TB	2.5	0.0	00:00:...	8	3	329	RuntimeBroker
7104	NT AUTHORITY\LOCAL SE...	9.28MB	2.00TB	2.5	0.1	00:00:...	8	9	266	WmiPrvSE
4748	NT AUTHORITY\SYSTEM	6.47MB	2.00TB	1.3	0.0	00:00:...	8	4	155	conhost
8612	NT AUTHORITY\SYSTEM	169.45MB	4.58GB	1.3	0.9	00:00:...	8	13	187	cribl
4404	EC2AMAZ-0P3VQFI\Admin...	4.28MB	2.00TB	1.3	0.0	00:00:...	13	9	411	ctfmon
1144	EC2AMAZ-0P3VQFI\Admin...	244.70MB	2.01TB	1.3	1.3	01:11:...	8	70	1569	firefox
0		56.00KB	8.00KB	1.3	0.0		0	4	0	Idle
404	NT AUTHORITY\NETWORK...	70.10MB	2.00TB	1.3	0.3	00:01:...	8	38	855	svchost

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

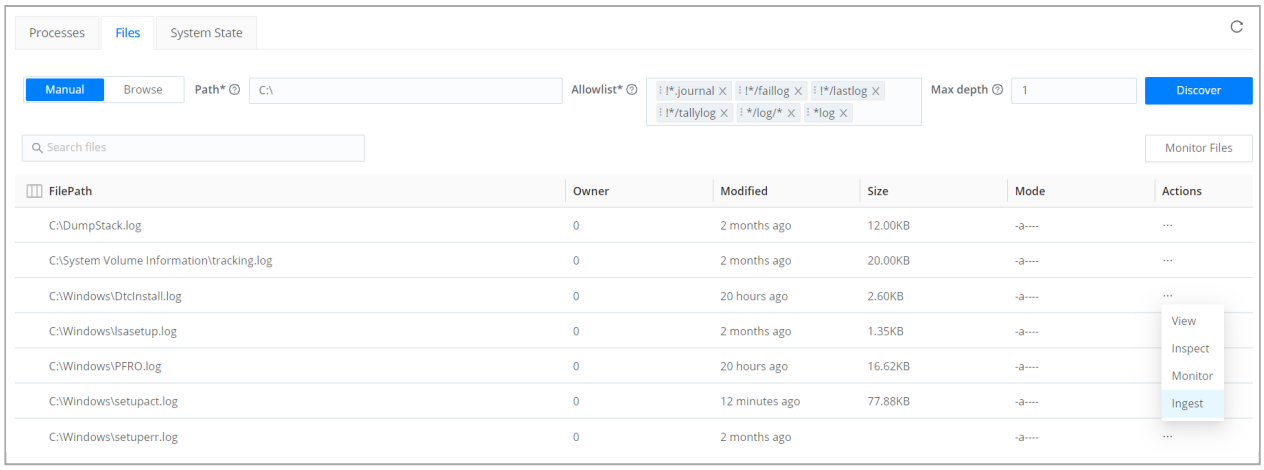
In this modal, click **All details** to get a table view of processes' information. To access this information, you would normally need to SSH to the machine; this view makes troubleshooting across multiple systems much easier.



All detail link

Files

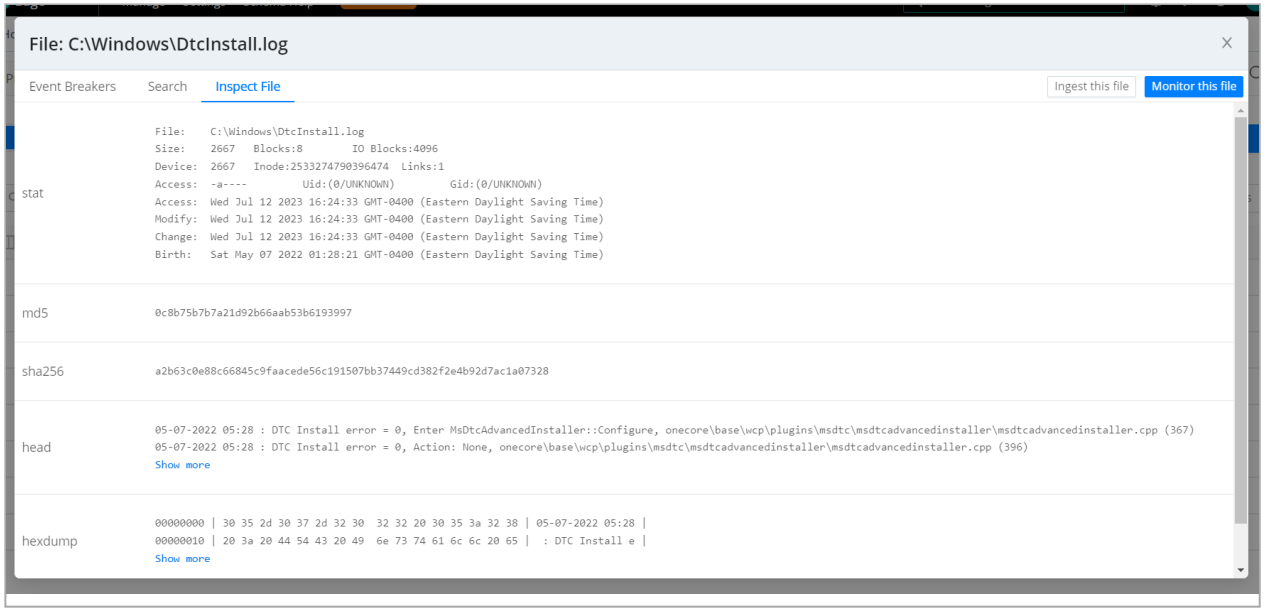
The **Files** tab enables you to specify a list of directories and files to actively monitor.



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:

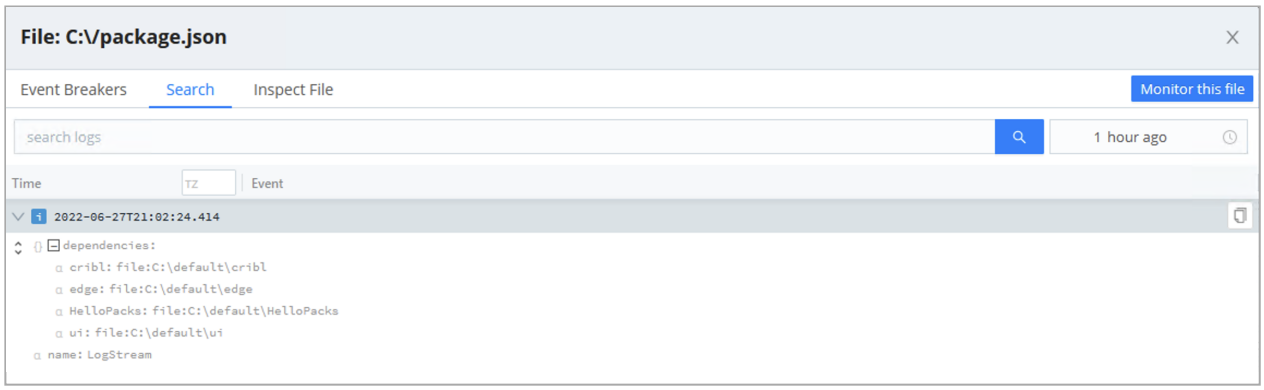
Path

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

Allowlist

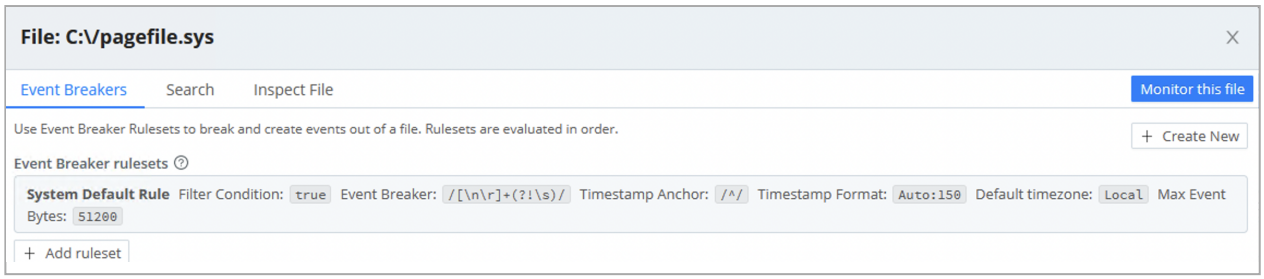
The **Allowlist** field supports wildcard syntax, and supports the exclamation mark (`!`) for negation.
For example, you can use `!*cribl*access.log` to prevent Cribl Edge from discovering its own access log.
The default filters are `*/log/*` and `*log`.

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.

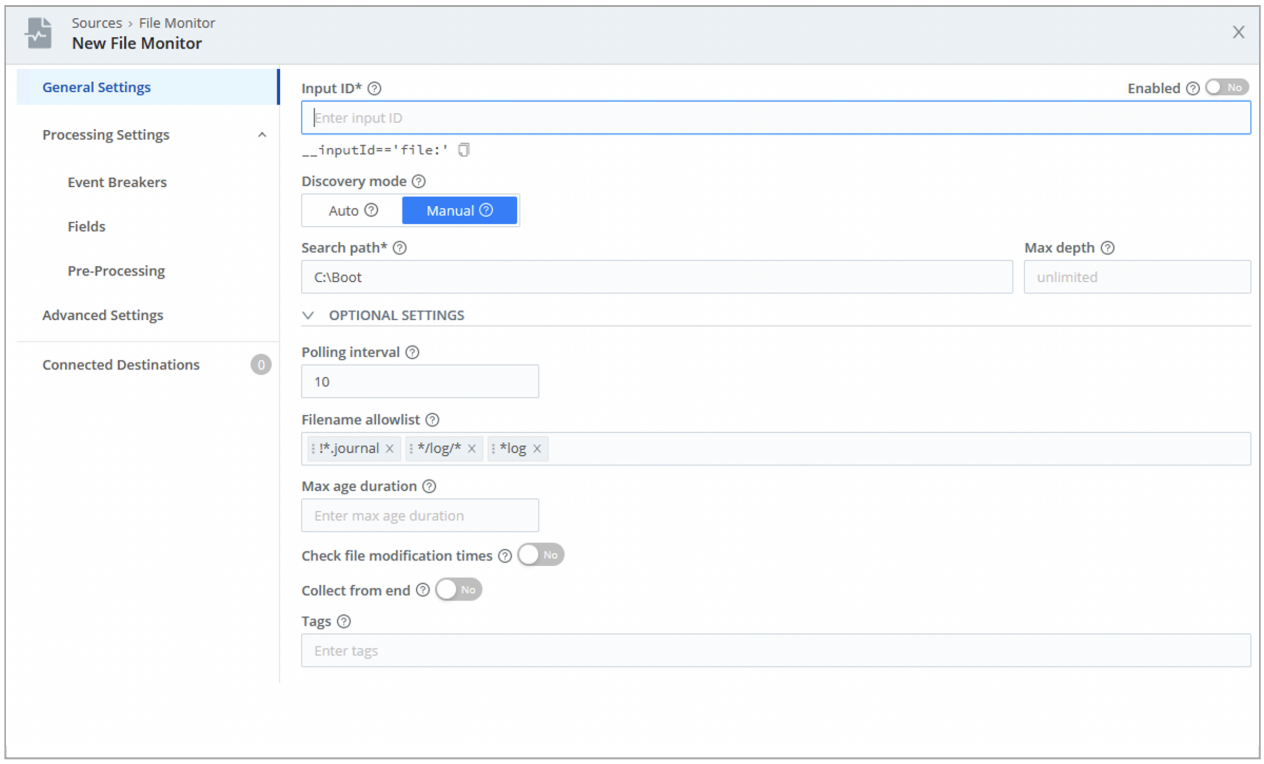
If you enter `0`, Cribl Edge will discover only the top-level files within the specified path. If you specify `1`, Cribl Edge will discover files one level down from the top. Follow this pattern to specify the depth you want.

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.

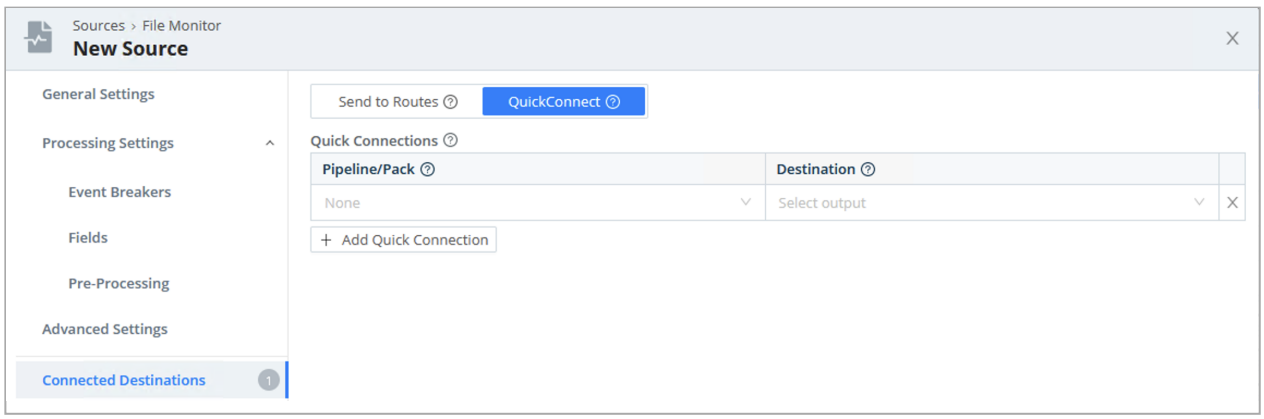


New File Monitor modal

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

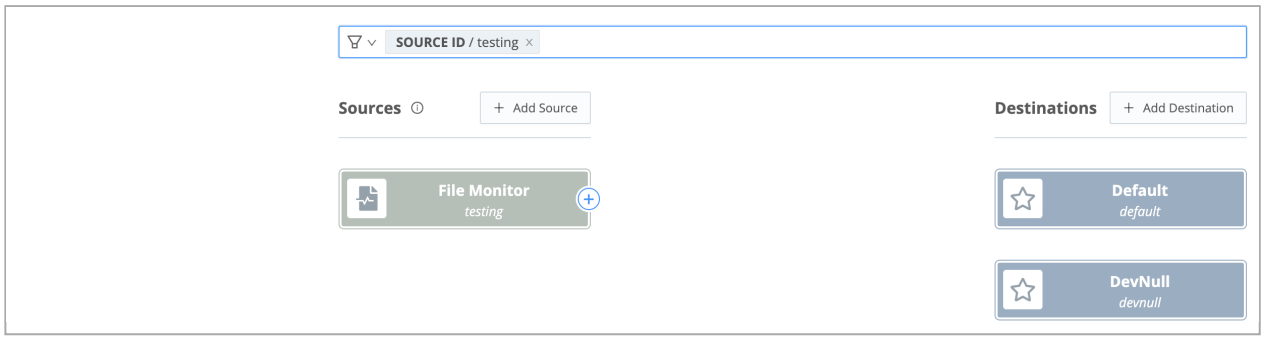
- **Search path**
- **Discovery mode**: Defaults to **Manual** for Windows.
- **Max depth**
- **Filename allowlist**

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.



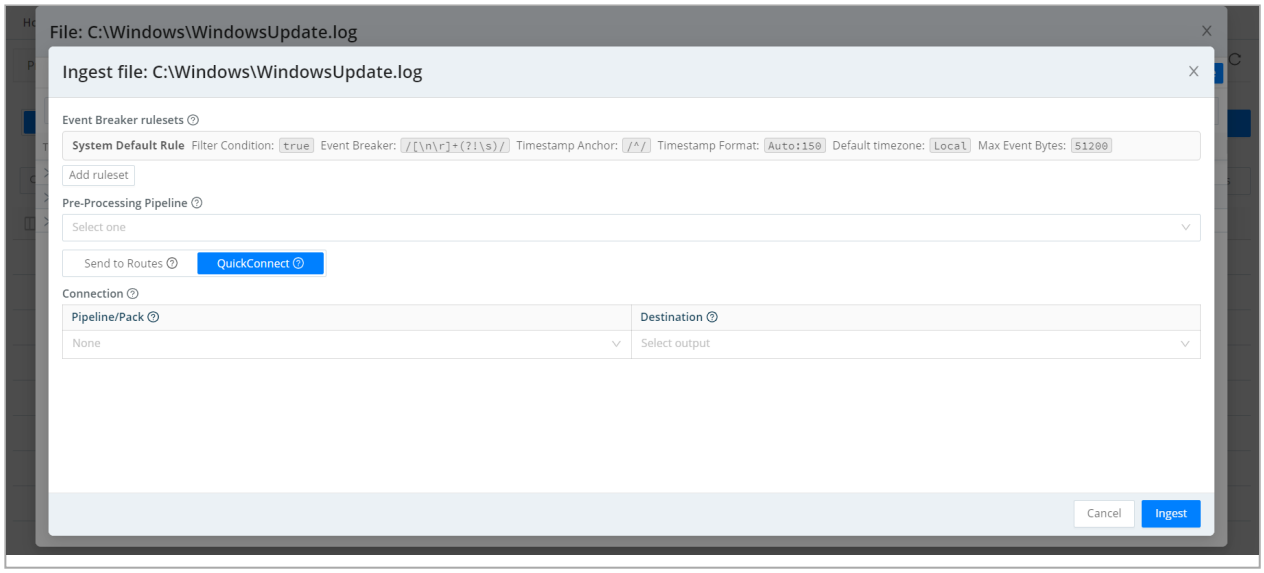
Collect page

Ingesting a File

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

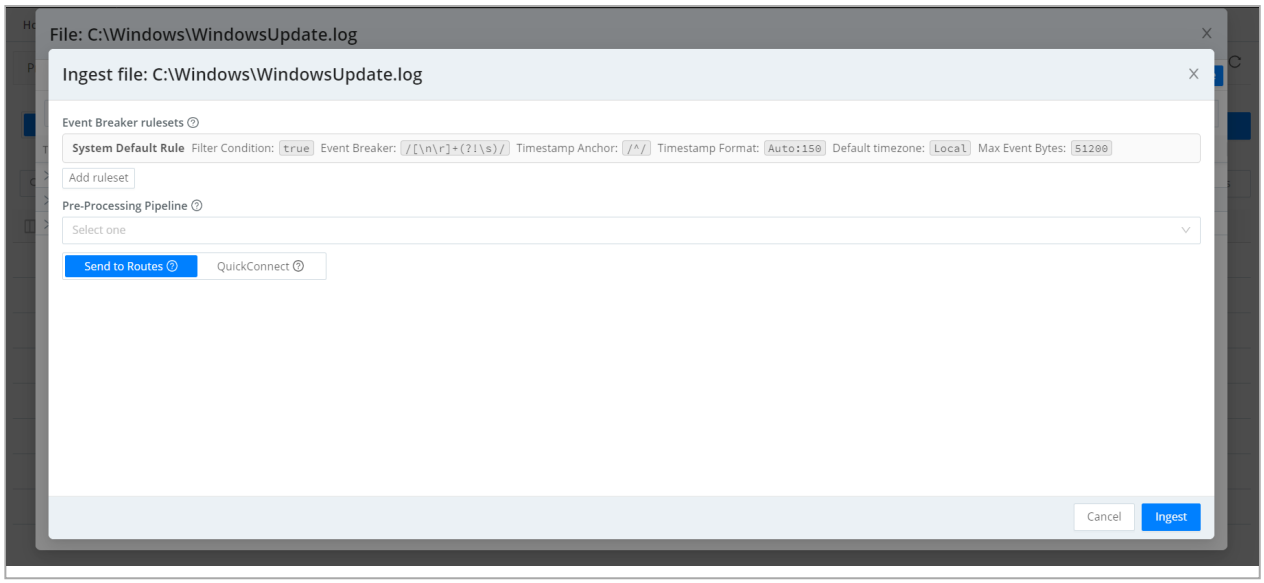
You have two options:

- 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:

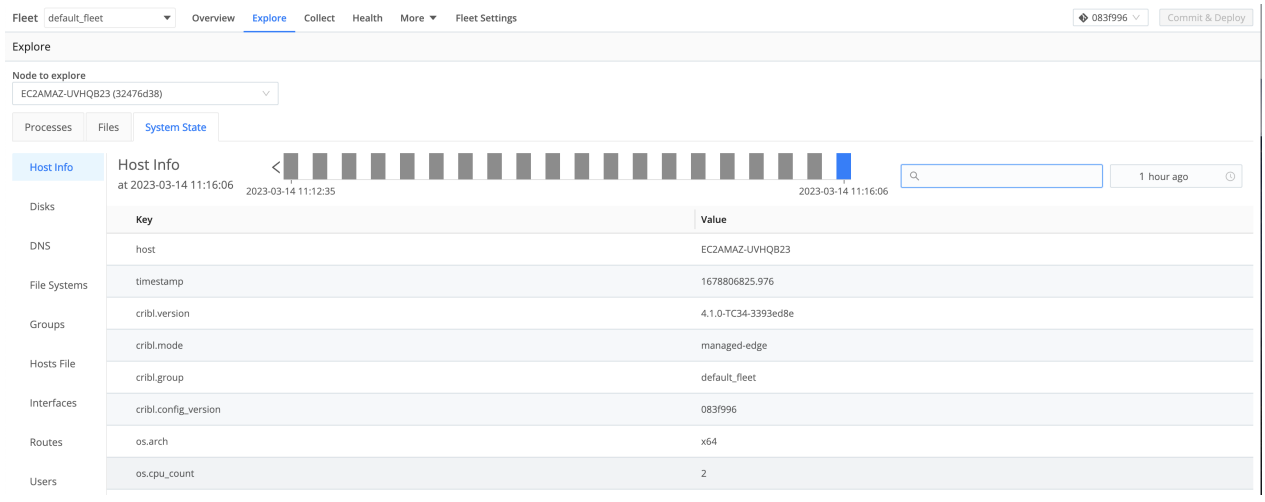
- Host Info

- 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.



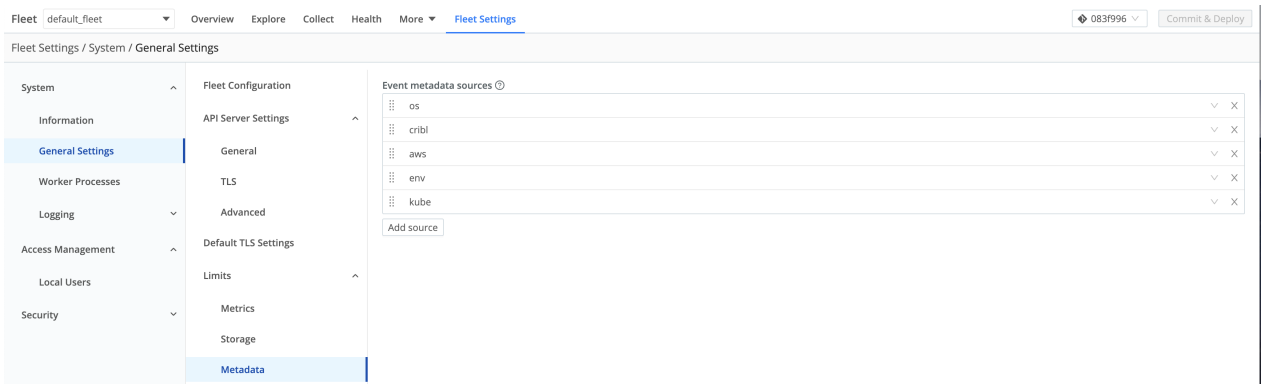
Host info / Metadata collected

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, 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.

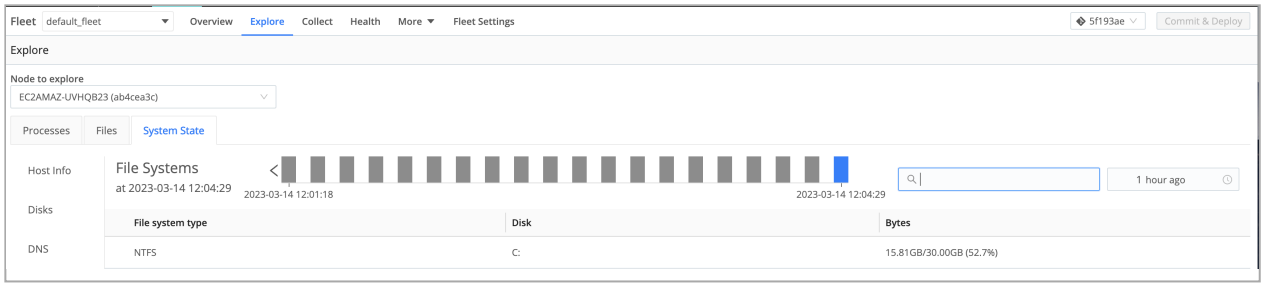
Disks

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

Disks tab

DNS

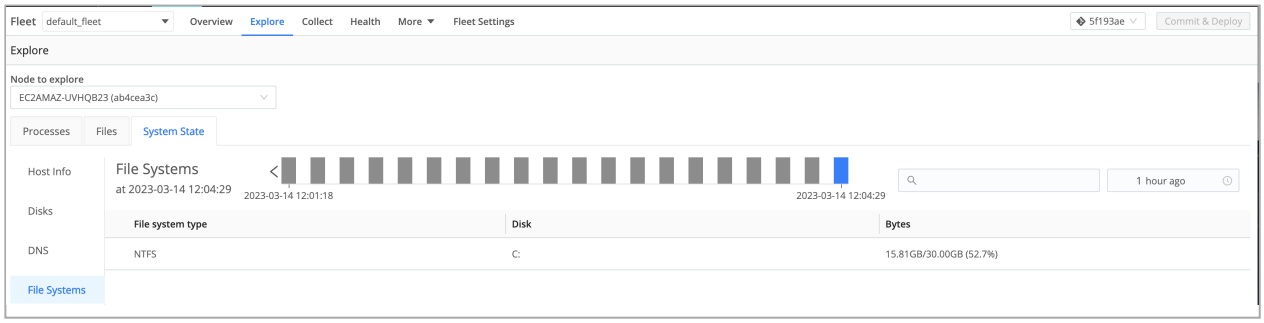
The **DNS** tab lists the DNS resolvers and search entries on the host system.



DNS tab

File Systems

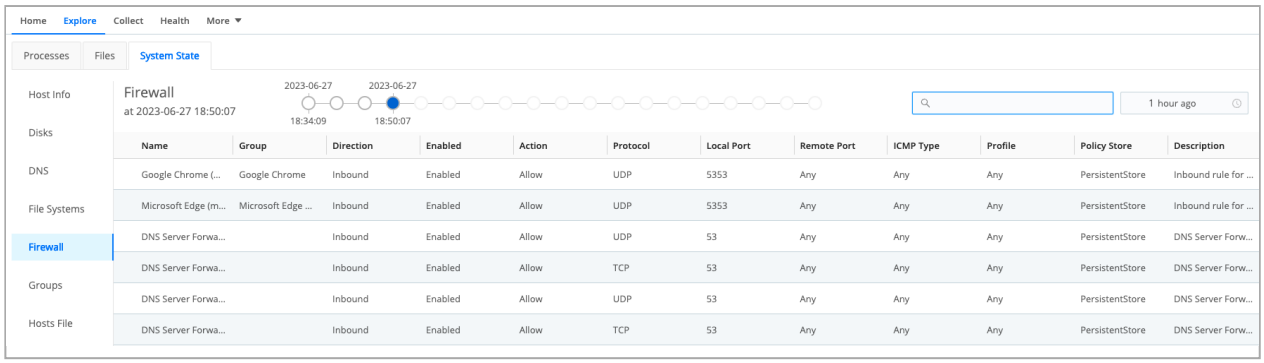
The **File Systems** tab displays an inventory of the mounted file systems on the host system.



File Systems tab

Firewall

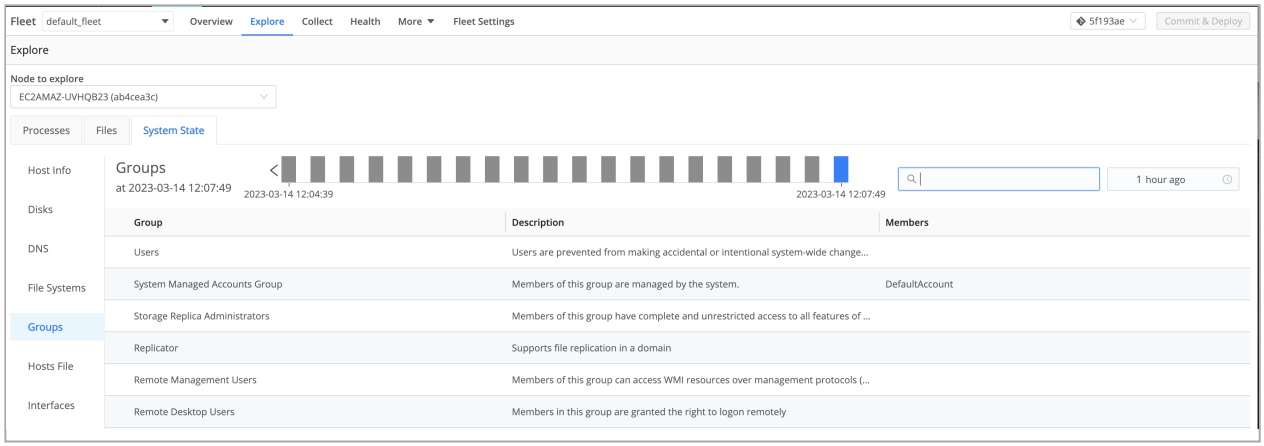
The **Firewall** tab displays a list of the host's defined firewall rules.



Firewall tab

Groups

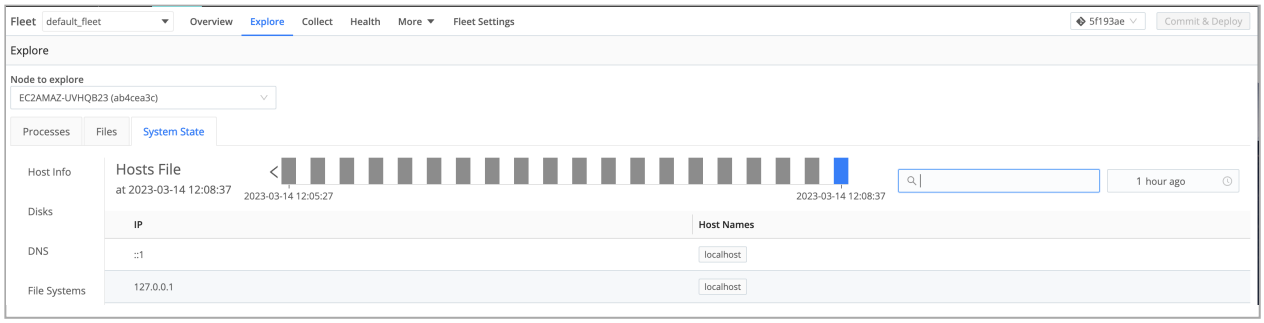
The **Groups** tab displays a list of local groups including their names, descriptions, and members on the host system.



Groups tab

Hosts File

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



Hosts File tab

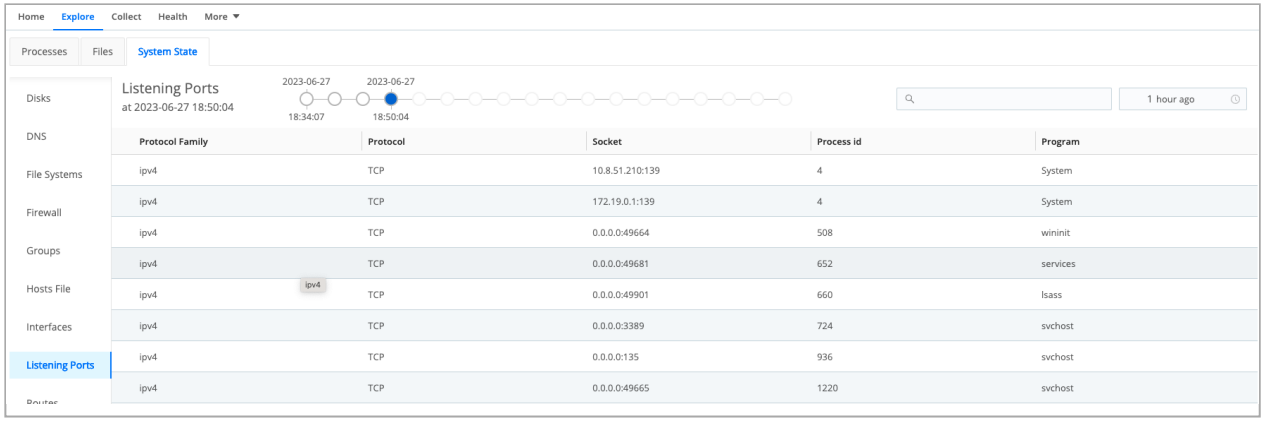
Interfaces

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

Interfaces tab

Listening Ports

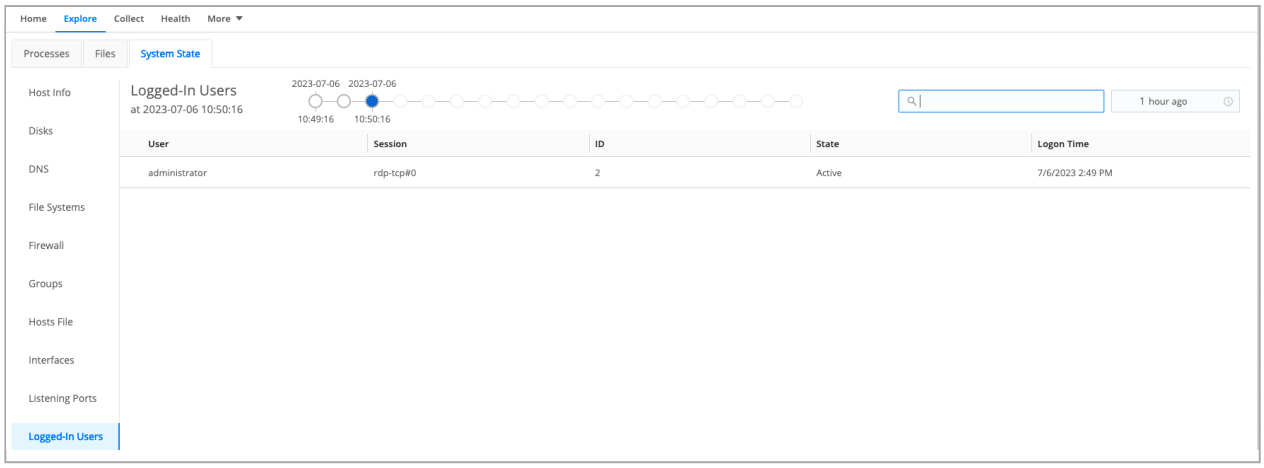
The **Listening Ports** tab displays a list of listening ports and their associated process identifier (pid).



Listening Ports tab

Logged-In Users

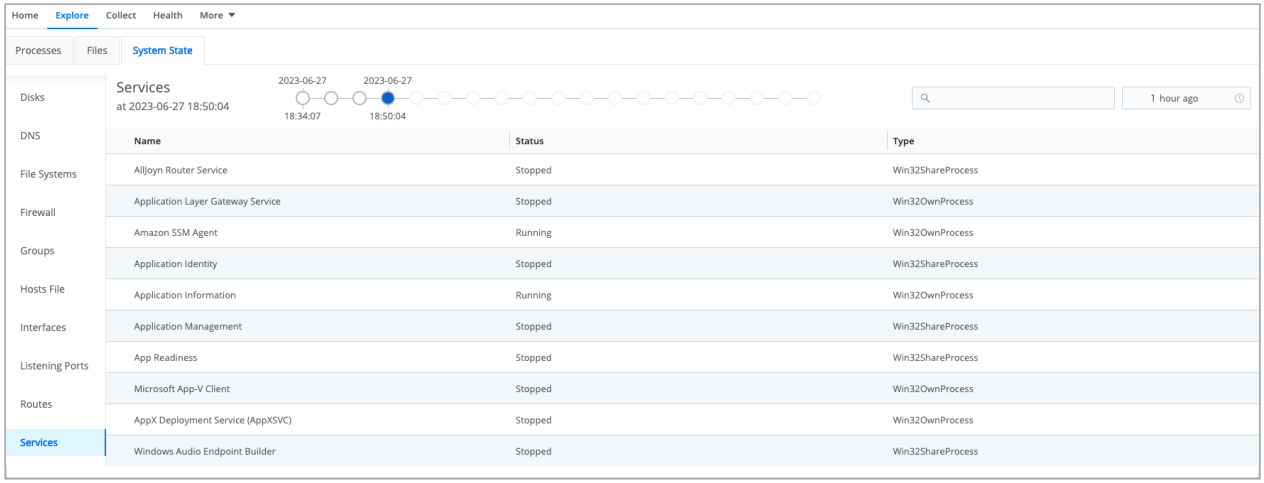
The **Logged-In Users** tab displays a list of currently logged-in users on the host.



Logged-In Users tab

Services

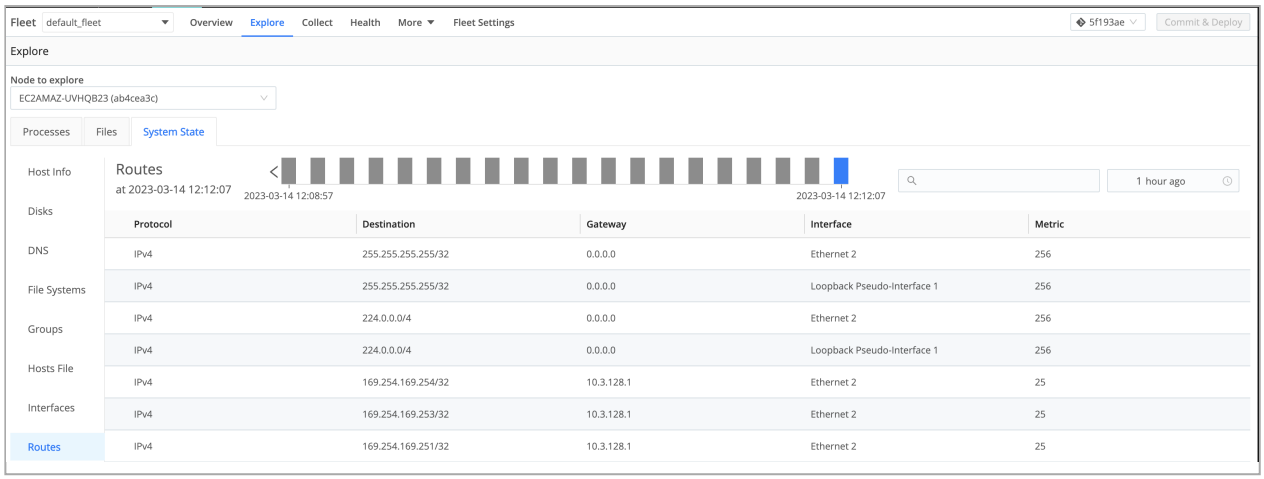
The **Services** tab displays a list of each configured service along with their running status.



Services tab

Routes

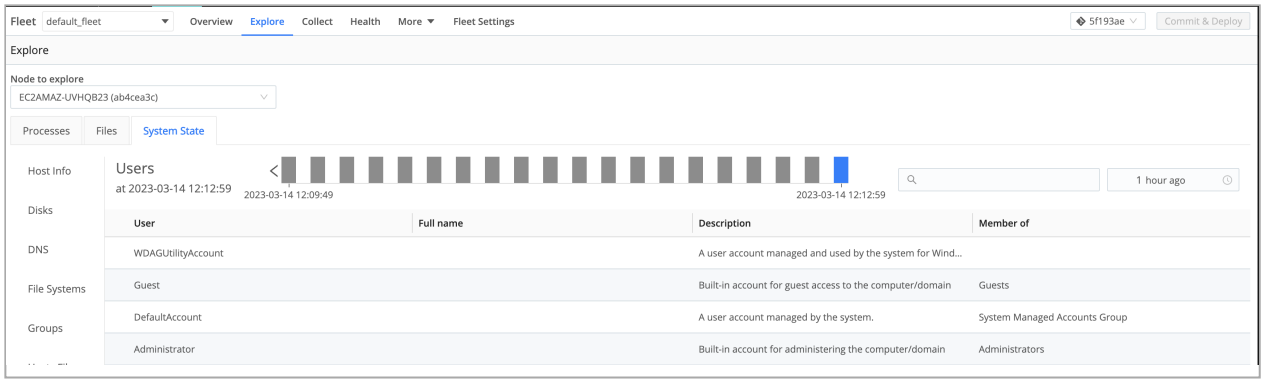
The **Routes** tab displays entries from network routes on the host system.



Hosts Routes tab

Users

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



Users tab

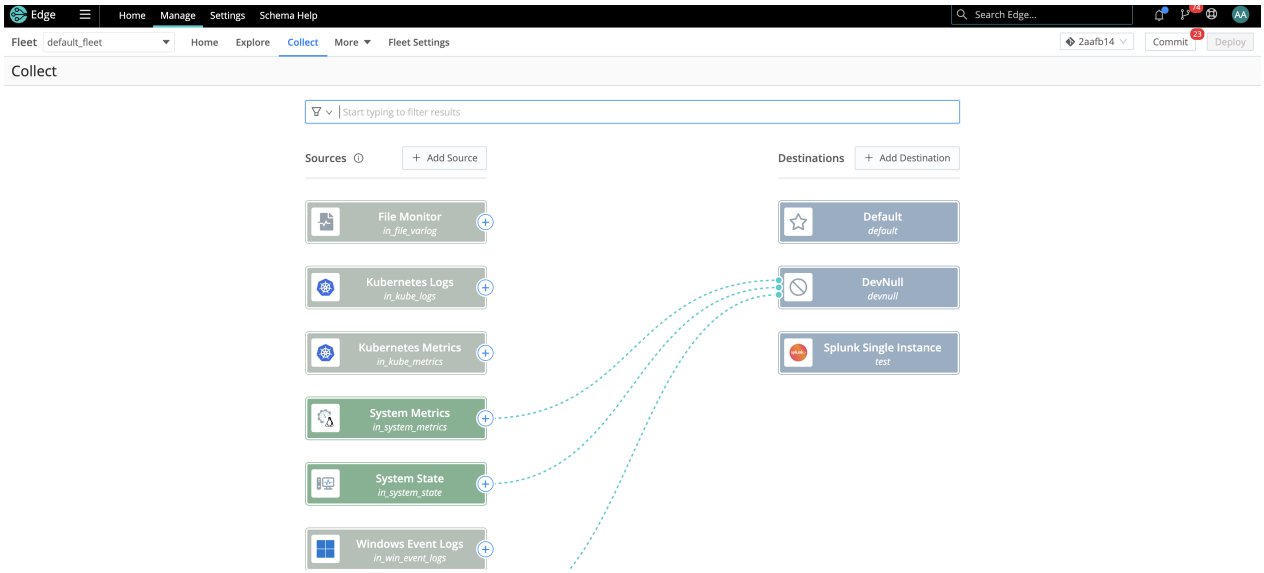
> Cribl University offers a course titled Collecting Data in Windows that provides a good overview of working with Windows. To follow the direct course link, first log into your Cribl University account. (To create an account, click the **Sign up** link. You'll need to click through a short **Terms & 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.

1.5. 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**.

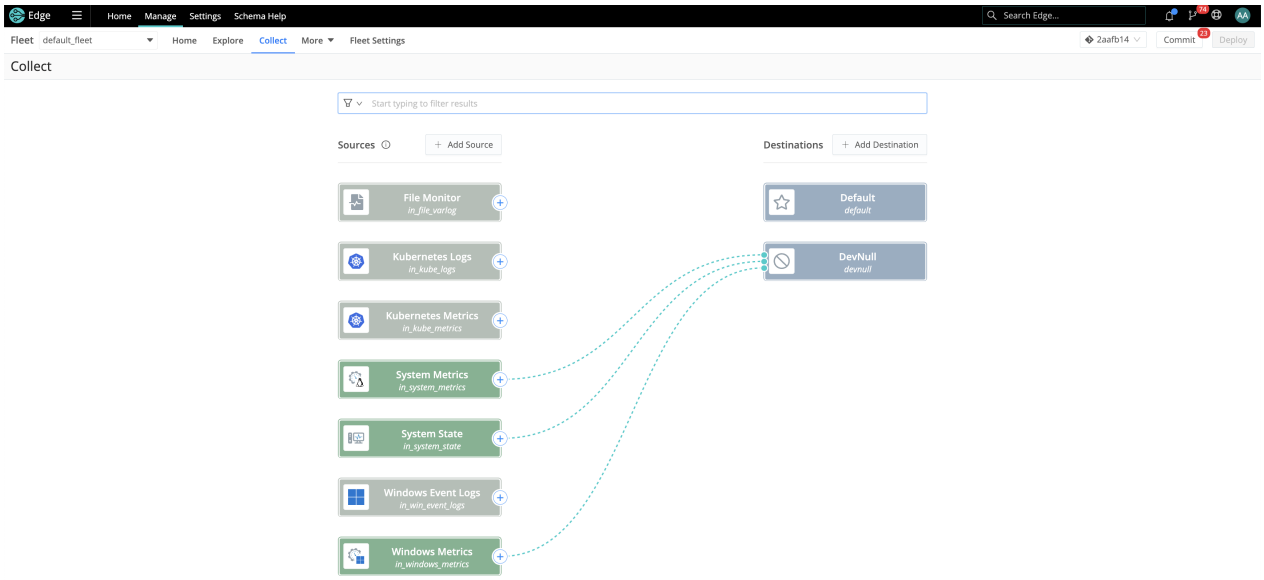
QuickConnect Versus Routes

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:



QuickConnect UI

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**.

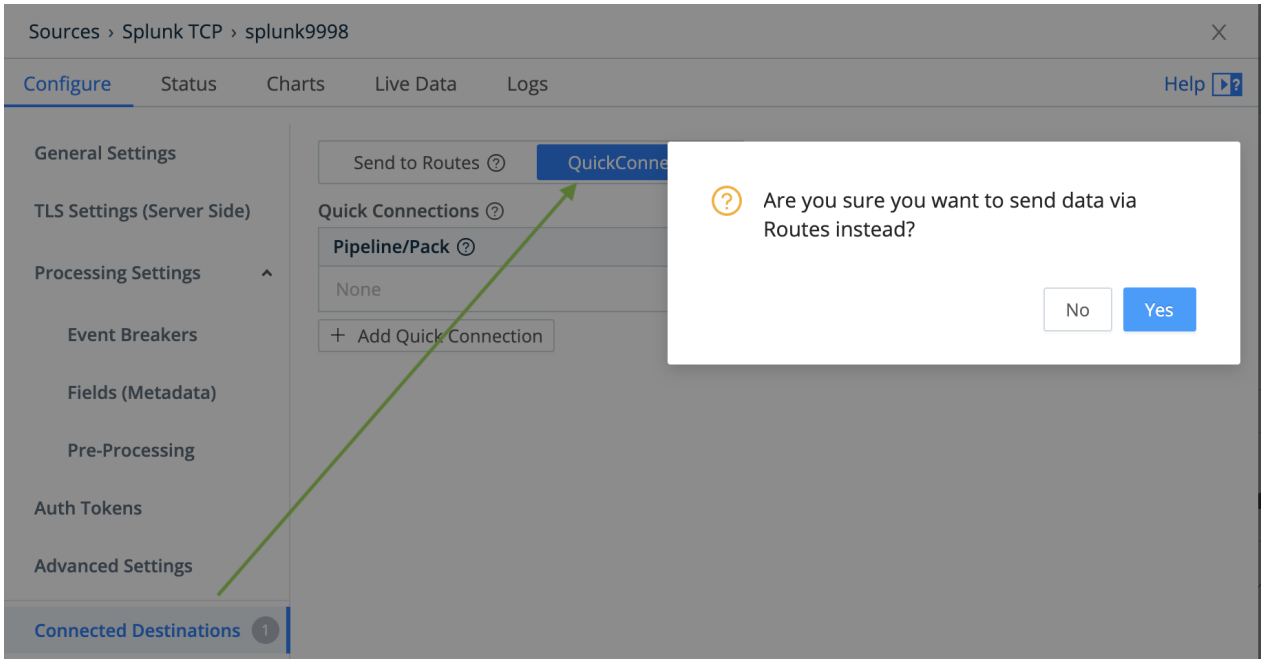
Switching Contexts

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.

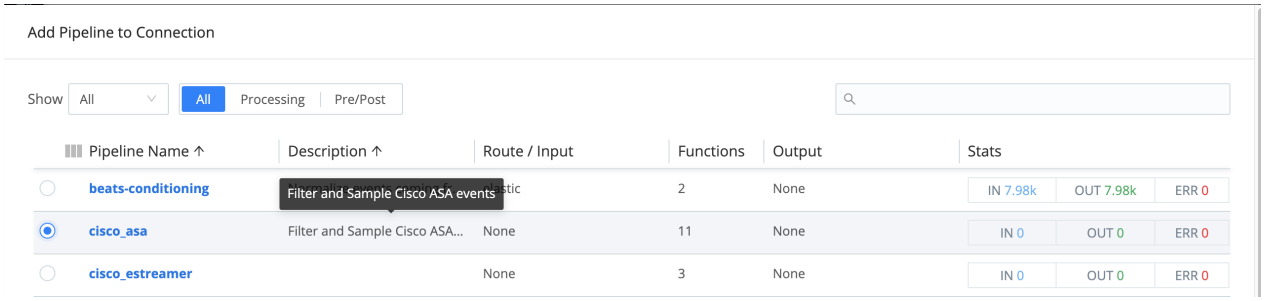
You can also go the opposite way: move a Source originally configured in QuickConnect to the Routing interface. In the Source's drawer or config modal, select the **Connected Destinations** left tab, and then click the resulting tab's **Send to Routes** button. Here again, you'll need to confirm your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

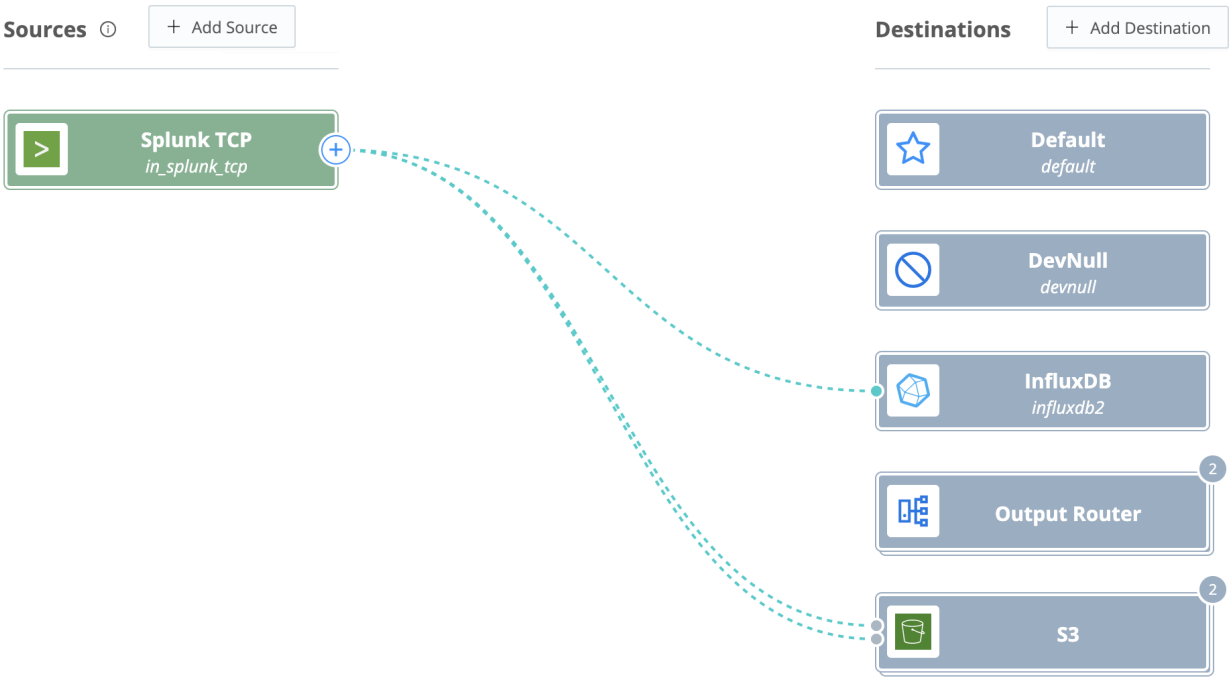
Here, you're selecting among already-configured Pipelines or Packs. **Resist the temptation to click on a bold blue Pipeline or Pack name.** (Doing so will open the Pipeline's or Pack's config in a separate browser tab, which might offer more configuration than you bargained for.)

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:



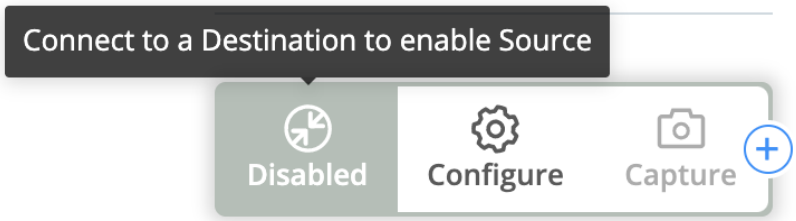
Multiple connections out, multiple connections in

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.

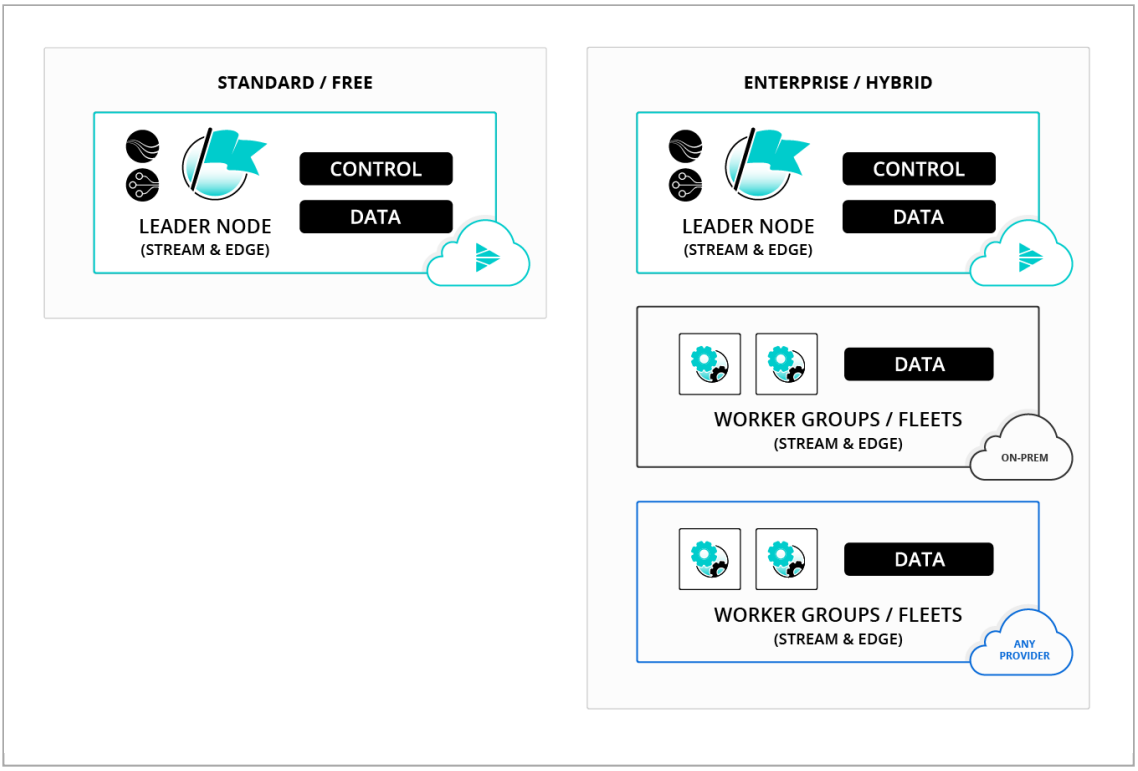
You can drag multiple connection lines from, or to, the same stack. If you drag or modify a connection from or to a stacked group, the UI will prompt you whether to similarly expand the group to access individual tiles.

2. Cribl.Cloud

2.1. Launch Guide

The fast alternative to downloading and self-hosting Cribl Edge software is to launch Cribl.Cloud. This SaaS version, whether free or paid, places the Leader and the Edge Node in Cribl.Cloud, where Cribl assumes responsibility for managing the infrastructure.

By upgrading to a Cribl.Cloud Enterprise plan, you can implement a hybrid deployment of any complexity. In hybrid deployments, the Leader (the control plane) resides in Cribl.Cloud, while the Workers that process the data (the data plane) can reside in any combination of Cribl-managed Workers/Edge Nodes, on-prem or private cloud instances that you manage, and 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? (Advantages)

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):

- Tap Cribl Edge's power, with no responsibility to install or manage software. Cribl.Cloud is fully hosted and managed by Cribl. so you can launch a configured instance within minutes.
- Automated delivery of upgrades and new features.
- Encrypted data at rest (configuration, sample files, etc.) at the disk level for Leader and Cribl-managed Worker instances.
- Free, up to 1 TB/day of data throughput (data ingress + egress) for all new accounts.
- Quickly expand your Cribl.Cloud deployment beyond the free tier's limits by purchasing credits toward metered billing. Pay only for what you use.

Getting Started

Your first step 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, a network and access boundary that isolates your Cribl resources from all other users. Each Cribl.Cloud account provisions a separate AWS account. Your instances of Cribl Stream, Cribl Edge, and Cribl Search are deployed inside a virtual private cloud (VPC) in this account.

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 click the **Go Enterprise** button at the top of your Portal.

> As of November 2022, 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.
>
> As of June 2022, Cribl completed its SOC 2 (Service Organization Control 2) Type II security compliance attestation.

About Cribl Edge (and This Document)

If you're new to Cribl Edge, please see our Basic Concepts page and Getting Started Guide for orientation. The current topic focuses on a Cloud deployment's differences from other deployment options – referred to below as "Cribl Edge binaries" or "customer-managed deployments."

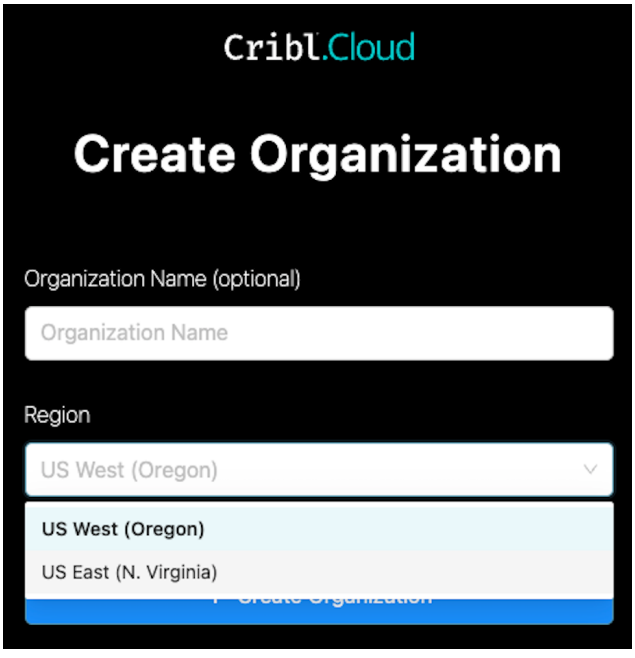
Cribl.Cloud always runs in distributed mode – see Simplified Distributed Architecture below for details.

Registering a Cribl.Cloud Portal

Ready to take the red pill? The next few sections explain 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. Select the **New User? Free signup** option, and register with your work email address.
3. Use the verification code from Cribl's email to confirm your registration.
4. 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).
5. Select an AWS Region to host your Cribl.Cloud Leader and Cribl-managed Workers. Cribl currently supports either the `US West (Oregon)` or `US East (Virginia)` Region.
6. Bookmark your Cribl.Cloud portal page, for all that follows.



Create Organization page – selecting a host Region

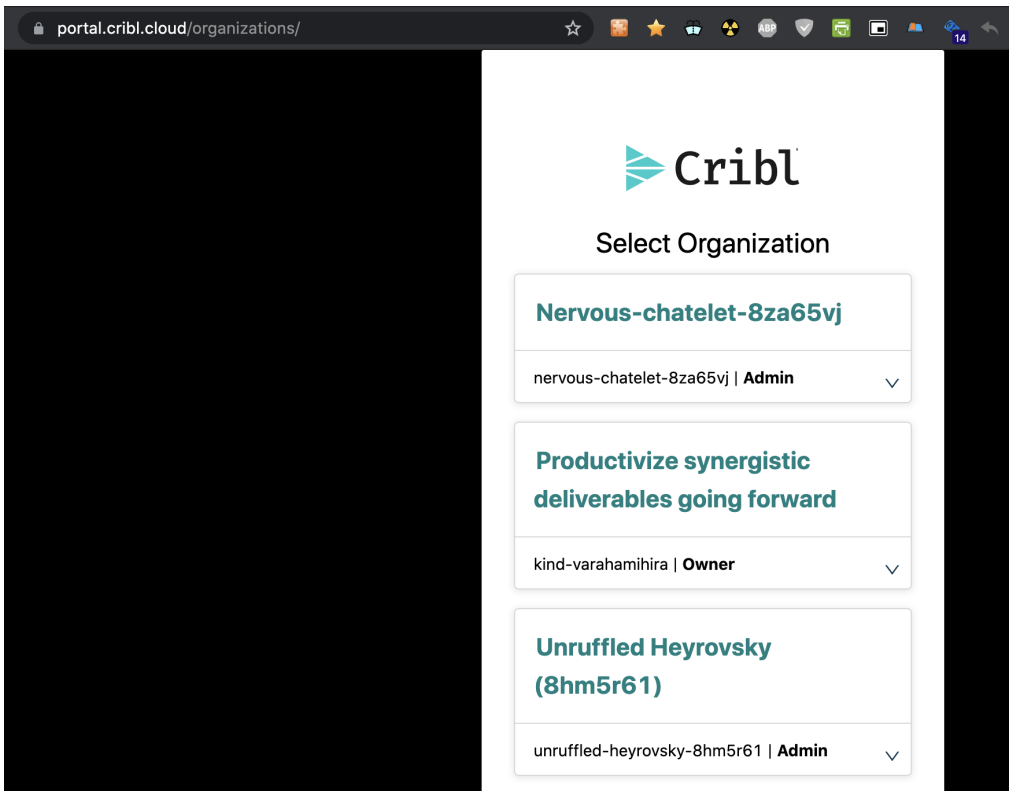
Note that each user can register only one active Organization. For details, see notes on Member Permissions.

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, click 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.

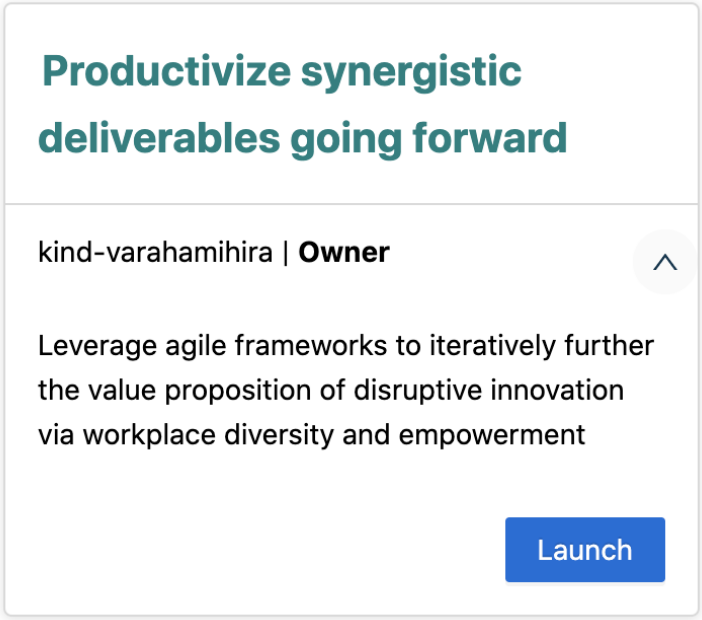
Select Organization Page

When you own or are a member of multiple Cribl.Cloud Organizations, the **Select Organization** splash page – displayed after you sign in – enables you to select which Organization you want to work with.



Select Organization interstitial page

Click any tile's \/ accordion to reveal a detailed description, if provided. Click the appropriate tile (or its open accordion's **Dashboard** button) to configure that Organization.



Organization tile's details and controls

You can click **Leave** if you want to remove yourself as a member of another owner's Organization. This option requires confirmation – proceed only if you're sure! (You won't see this button on Organizations that you own.)

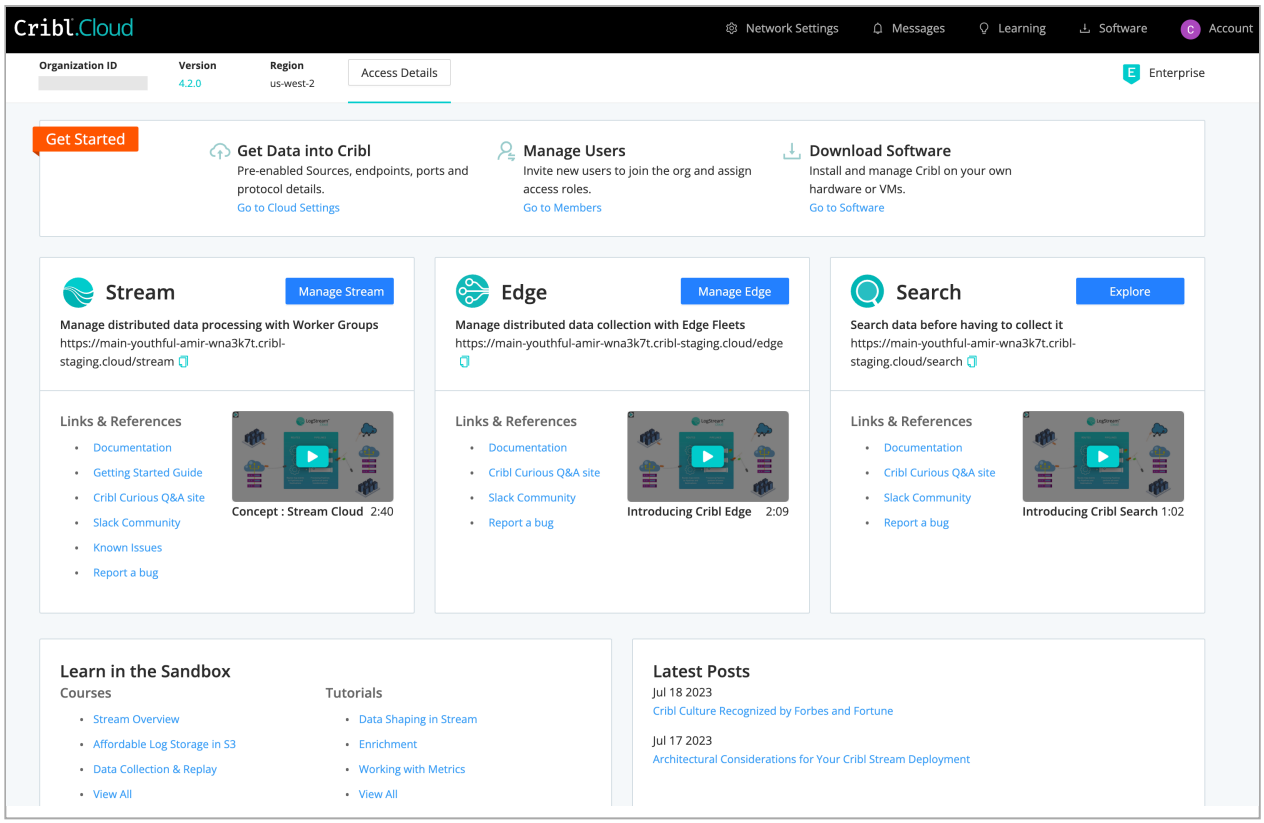
Exploring the Cribl.Cloud Portal

Now that you're here – explore the furniture. The Cribl.Cloud portal's top navigation allows you to navigate among the following pages/links:

- Portal (**Cribl.Cloud** logo)
- Messages
- Learning
- Software
- Account (including Organization details)

Portal Page (Cribl.Cloud Logo)

When you log into the Cribl.Cloud portal, you'll land here. The main events here are the **Manage Stream**, **Manage Edge**, and **Explore [Search]** buttons. Click these to launch (respectively) Cribl Stream, Cribl Edge, or Cribl Search in a new tab.



Cribl.Cloud portal

However, the surrounding page offers lots more useful information:

- On the page body, you'll find links to multiple Cribl resources – documentation, support (Community Slack and bug reporting), free Sandbox training, and blog posts.
- In the Overview strip just below the top black menu, you'll find detailed configuration information about your Cloud Organization.
- By clicking the top nav's **Network Settings** link, you can check and manage connectivity details – data Sources, access control, and trust relationships – for your Cribl-managed Cloud Workers.

Overview Strip and Access Details (Group Details) Modal

From left to right, this upper strip displays the following config details:

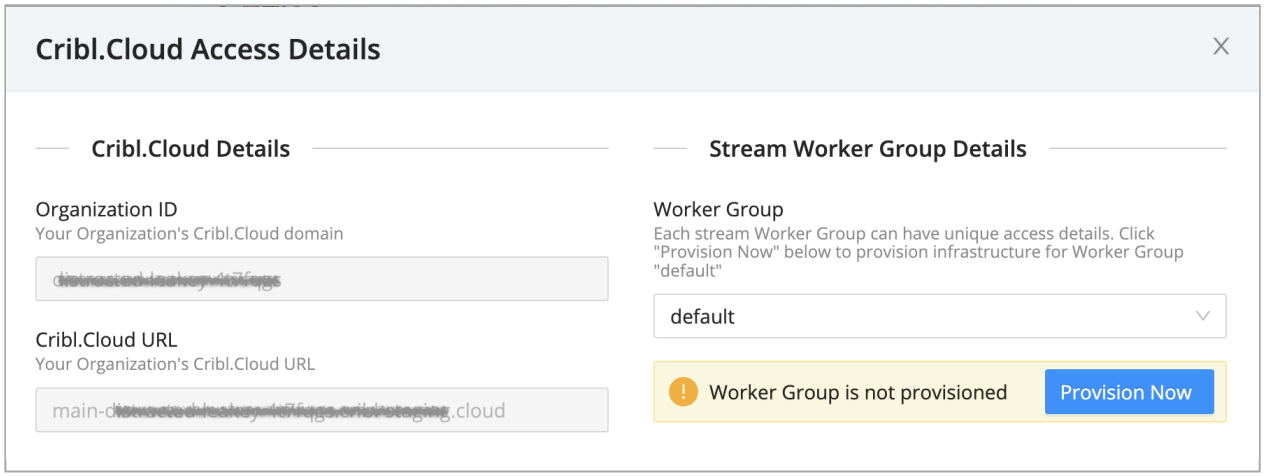
Org ID: Domain at which you access the associated Cribl.Cloud Organization.

Version: The version of Cribl Stream/Edge applications deployed to your Organization and its Cribl-managed Workers.

Region: The AWS Region where you're running Cribl applications. (Cribl.Cloud currently supports either the **us-west-2** or **us-east-1** Region.)

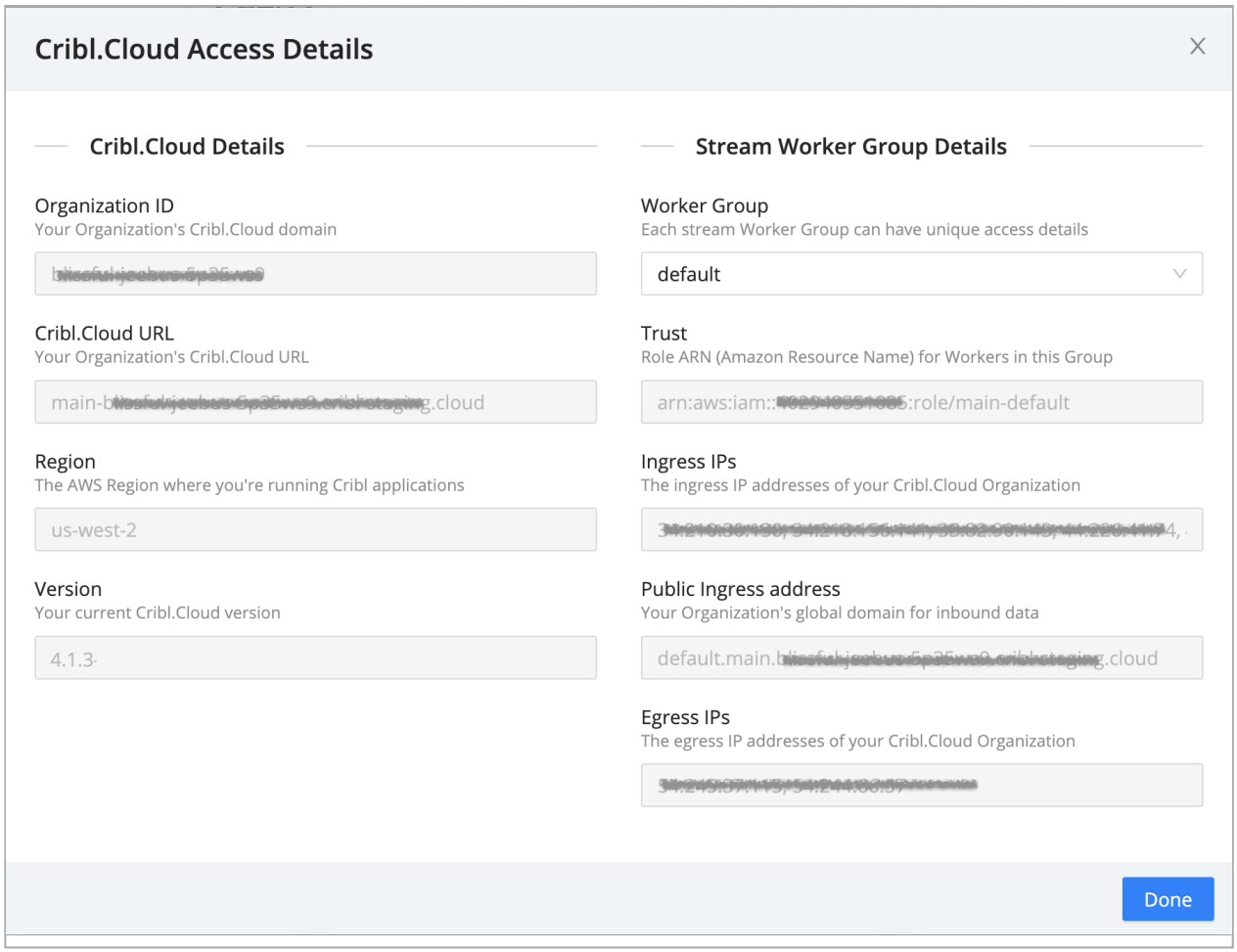
Access Details: Click this button to open a modal whose right column provides a consolidated, read-only display of the following **Stream Worker Group Details**. Some of these options are configured on different tabs, as noted below.

- **Worker Group**: Use this drop-down to select any Group of Cribl-managed Workers that you've configured (including `default`). The remaining fields on the right will display details specific to that Group.
- **Provision Now**: This button will replace all the fields listed below when a Group is dormant. Click the button when you're ready 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.



Access Details modal for an unprovisioned Group

- **Trust**: Role ARN for Workers in this Group. You configure these ARNs on the Trust Tab.
- **Ingress IPs**: The IPv4 ingress addresses associated with your Cribl.Cloud Organization's endpoints. These addresses will remain constant, so you can build firewall rules around these addresses.
- **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. Cribl.Cloud currently does not offer static egress IPs.



Access Details modal for a provisioned Group

> Configuring Stream Groups (beyond the `default` Group) requires an Enterprise plan. For details about creating and provisioning Groups, see Cribl.Cloud Worker Groups.
>
> The **Access Details** modal's left side displays Organization-wide access details, including the **Cribl.Cloud URL** of your Org's Leader/control pane. You'd use this URL for certain API calls and certain Collection operations coordinated by the Leader. Use the right-side details to configure data flow through individual Groups.

Network Settings Page

Clicking the top nav's **Network Settings** link opens a page with connectivity details, spread across three upper tabs: **Data Sources**, **ACL**, and **Trust**.

Data Sources Tab

The **Data Sources** tab lists ports, protocols, and data ingestion inputs that are open and available to use. Use the **Group** drop-down to filter these details per Group of Cribl-managed Workers in the Stream app. Return to

this tab to copy Ingest Addresses (endpoints) as needed. For details, see Available Ports and TLS Configurations.

Trust Tab

The **Trust** tab provides **Worker ARN**s (Amazon Resource Names) that you can copy and paste 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 Workers.

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 Cribl-managed Workers. You cannot use this technique to enable access to hybrid Workers on customer-managed Cribl Edge instances.

ACL Tab

This **Access Control List** defines Rules (IPv4 CIDR ranges) to restrict data sent to your data sources. The Rules you define here are global to all your Cribl-managed Groups of Stream Workers.

The default `0.0.0.0/0` rule (modifiable) imposes no limits. Click `+` to add more rules, or click `X` to remove rules. 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`.

Click **Save** after adding, modifying, or removing rules. Each change takes up to 5 minutes to propagate. Cribl.Cloud will display an `ACL update in progress...` 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 your Cribl-managed Workers. You cannot use this technique to set access rules on hybrid Workers running in customer-managed Cribl Edge instances.

Cribl Edge UI Access

Clicking the **Manage Stream** or **Manage Edge** button opens (respectively) your Stream or Edge Leader in a new browser tab. All of the application's Cloud-supported features are available from this landing page.

Messages Drawer

Clicking the top nav's **Messages** link opens the **Message Center** right drawer. Here, you will find Cribl.Cloud status and update notifications from Cribl, with **Unread** messages above the **Read** group.

Learning Page

Clicking the top nav's **Learning** link opens the **Learning** page, which provides links to everything you need to learn about Cribl Edge in order to goat forth and do great things:

- Sandboxes (free, interactive tutorials on fully hosted integrations).
- Documentation.
- Product and plans overview (pricing comparison).
- Cribl events (including future and archived Webinars).
- Concept/demo videos.

Software Page

If you prefer to take the blue pill, this page offers download links for Cribl Edge, Cribl Edge, and AppScope software. You can download either binaries or Docker containers (hosting Ubuntu 20.04), to install and manage on your own hardware or virtual machines.

Account Menu

This menu offers a self-explanatory **Sign Out** link, and an **Organization Selection** submenu (fly-out) that works like the Select Organization page: click its links to traverse to other Organizations. For an Organization's owner only, it also includes a link to the Organization page.

Account tab

Organization Page

Displayed only to an Organization's owner, this page offers Details, Members, and (where applicable) Billing and SSO tabs along its top.

Details Tab

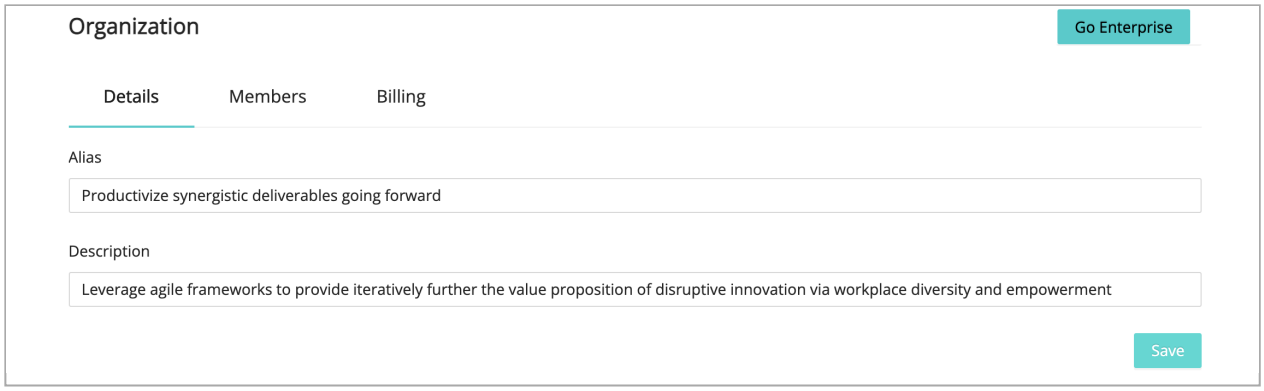
The **Organization** > **Details** tab offers these controls to make your Cribl.Cloud deployment more recognizable than its randomly generated **Organization ID** (displayed at the top):

Alias: Optionally, enter a "friendly" name for your Organization. Upon signing in, members will see this alias above the Organization ID on the Select Organization page.

Description: Optionally, use this field to add further details about your Organization. On the Select Organization page, members can view these details by expanding the Organization's tile.

Opt in to beta features: If displayed, this toggle 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 Cloud.

Click **Save** to immediately apply your changes.



Organization Details tab

Members Tab

The **Organization** > **Members** upper tab provides access to inviting and managing other users.

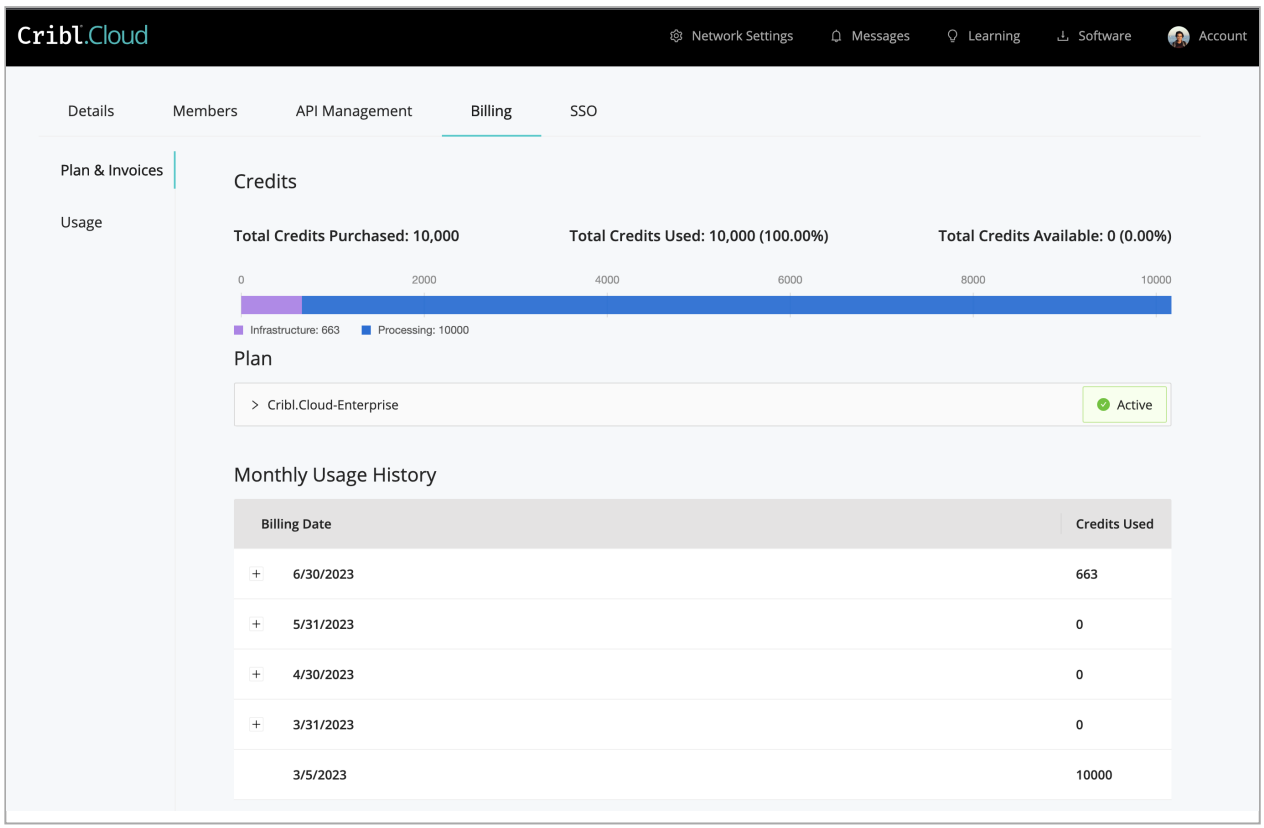
Billing Tab

The **Organization** > **Billing** upper tab is displayed only to owners of an Organization on a paid license plan. It provides Plan and Usage left tabs.

Plan & Invoices Tab

The **Plan & Invoices** left tab displays a mercury bar of purchased, used, and available **Credits** on your account. Color-coding breaks down usage by infrastructure versus processing (data throughput).

Below that is an expandable **Plan** details section. Expandable **Monthly Usage History** rows offer details about your credits usage in the current and prior months. Here, you can break out usage on Cribl Search, on hybrid Workers (ingest billing only), and on Groups of Cribl-managed Workers (with ingest versus infrastructure breakouts per Group).



Billing > Plan & Invoices tab

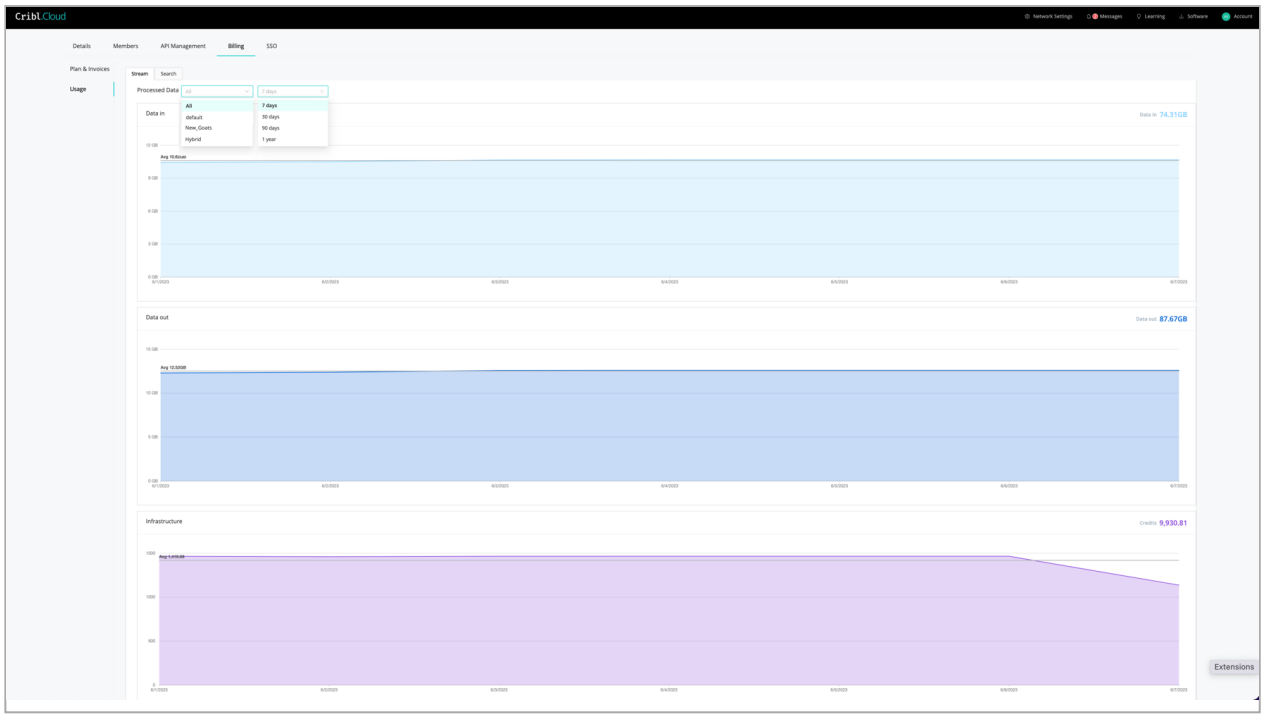
Credits carry over across billing periods, as long as you renew your Cribl.Cloud plan.

Usage Tab

The **Usage** left tab provides nested tabs for **Stream** versus **Search**. (If you're keeping tabs, this is a third level of selectable tabs.)

The **Stream** tab graphs credits usage for **Data in**, **Data out**, and **Infrastructure** over a selectable trailing period of 7 days to 1 year. 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. On the left drop-down, you can filter the aggregate display down to individual Cribl-managed Groups, or to all hybrid Groups.



Billing > Usage tab

The **Search** tab displays billed compute CPU hours. It provides a similar graph, with averages over time and a total per period. You'll find the same options for time granularity.

SSO Tab

This tab appears on an Enterprise plan, enabling you to configure federated authentication to your Cribl.Cloud Organization from an OIDC or SAML identity provider. For details, see Cribl.Cloud SSO Setup.

Managing Cribl.Cloud

Once you've registered on the portal, here's how to access Cribl.Cloud:

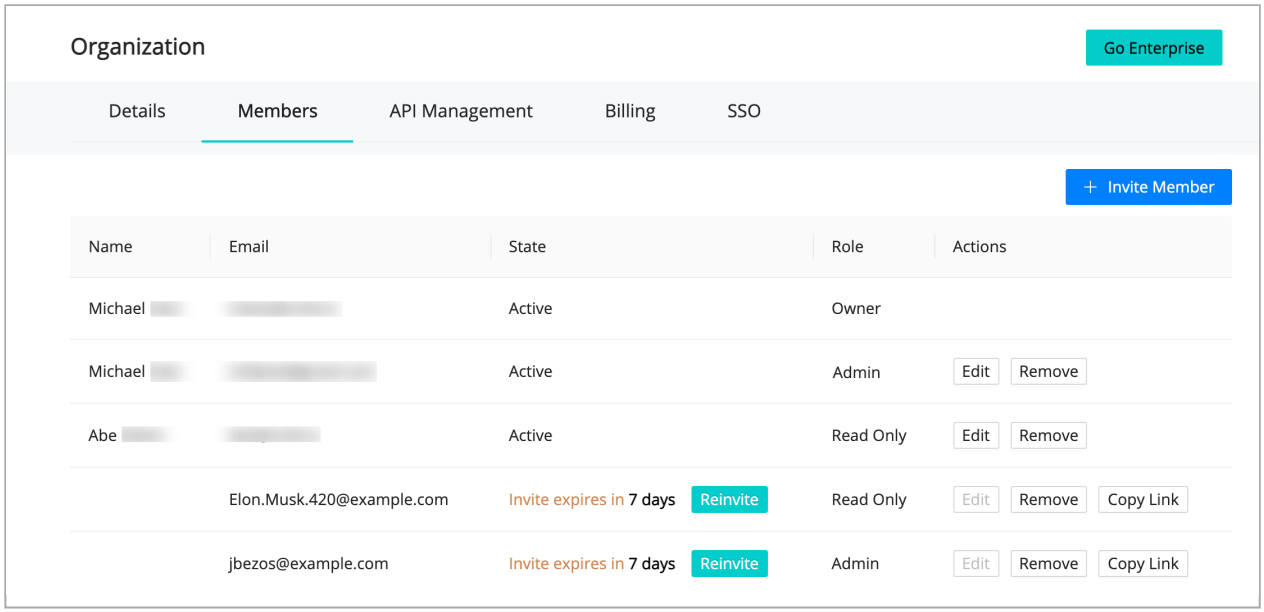
1. Sign in to your Cribl.Cloud portal page.
2. Select the Organization to work with.
3. From the portal page, select **Manage Stream**, **Manage Edge**, or **Explore [Search]**.
4. The selected application's UI will open in a new tab or window – ready to goat!

Note the **Cribl.Cloud** link at the Cribl.Cloud home page's upper left, under the **Welcome!** message. You can click this link to reopen the Cribl.Cloud portal page and all its resources.

Cribl.Cloud link takes you back to the portal

Inviting and Managing Other Users

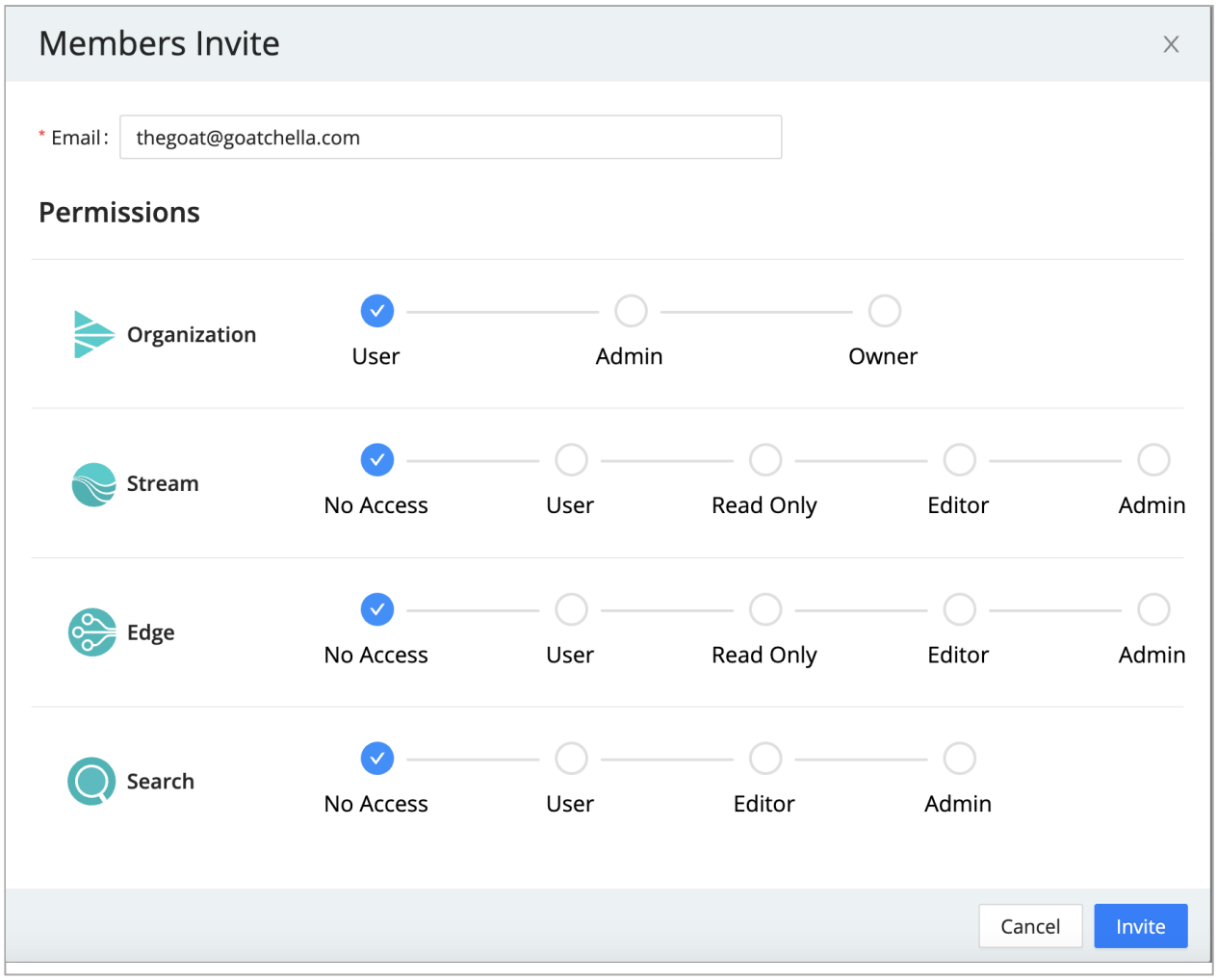
From the **Organization** > **Members** tab, an Organization's owner can invite new users to join the Organization, assign access Permissions to new and existing members, remove pending invites, and remove existing members.



Organization > Members tab: Managing Invites and Members

Inviting Members

Click **Invite Member** to open the modal shown below. Enter the **Email address** of the new user you want to invite, assign them appropriate **Permission** on your Organization (explained just below), and then click **Invite** to send the invitation.



Invite User modal

Organization Member Permissions

Newly configured Members start out with the permissions they were granted when they were invited at the Organization level, as well as the self-explanatory **No Access** Permission on each product. An Organization's Owner can assign the following Permissions at the Organization level.

> For Permissions that you can assign at the lower product, Fleet, and resource levels (with an Enterprise plan), see our Members and Permissions topic, which also covers on-prem counterparts to the Cribl.Cloud Organization-level Permissions listed here.

PERMISSION	DESCRIPTION
User	The lowest level of access. At the Organization level, gets the Member into the system, allows them to update their own Member profile, and allows them to view (but not configure) any Stream Worker Groups they have been given access to.

PERMISSION	DESCRIPTION
Admin	This Permission grants the `Read Only` Permission on all Products, plus the ability to view and execute Leader commits, and the ability to view and modify Global Settings.
Owner	This is a superuser Permission. An Owner can manage Organization Members, update their own Member profile, manage ACLs (Access Control Lists), view and update SSO settings, view Data Sources and Trust Policies, view and update Organization Details, delete an Organization, manage API Credentials, view and manage Cribl Suite Global Settings, view and manage (provision, update, and delete) Stream Worker Groups, view and execute Leader commits, and view the Org's Billing dashboards. Owners also get **Admin** access on all products.

Note that:

- Without an Enterprise plan, a Cribl.Cloud Organization's Owner can assign only the `Owner` or `Admin` Permission to other users. (For all available Enterprise features, see Pricing.)
- An Enterprise plan is also required for expanded Permission-based access control – i.e., the ability to manage Permissions on individual products, Fleets, and resources.
- Only an Organization's Owner can manage the Organization's Details tab.
- An Organization's Owner Permission can be shared among multiple users. (Among other use cases, this provides a way to gradually transfer primary ownership.) Also, a user can be invited to the Owner Permission on multiple Organizations.
- 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.
- You assign Permissions per individual user, when you invite them to your Organzation. Cribl.Cloud does not currently support globally predefining or assigning group Permissions, as with on-prem Cribl Edge. However, Owners can change users' Permissions after those users join their 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.

Responding to Invites

At the address you entered, the new 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.

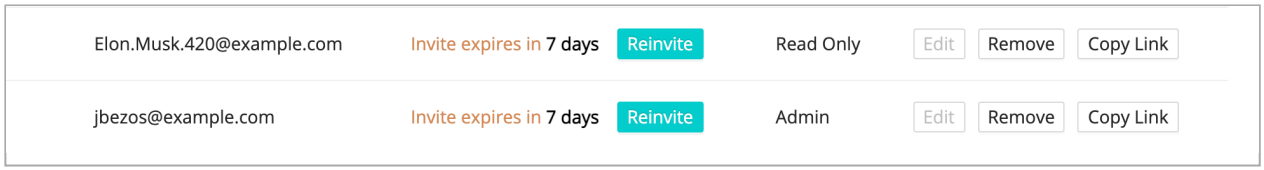
After signing in, they'll have access to your Organization and Cribl Edge instance at the Permission level you've specified.

Managing Invites

While an invite is pending, the **Organization** > **Members** tab offers you these options to deal with commonly encountered issues:

- **Reinvite**: If your invited member didn't receive your invitation email, you can click this button to resend it.
- **Copy 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.
- **Remove**: 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 7 days, if an invite has been neither accepted nor revoked, it expires. In this case, it is removed from the **Members** tab.



Managing Invites

Managing Members

Once a user has accepted an invite, the **Organization** > **Members** tab offers you these options to modify their membership in your Organization:

- **Edit**: Switch this member to a different Permission. (The **Edit** option is displayed only if you have an Enterprise plan.)
- **Remove**: Remove this member from your Organization. After clicking this button, you'll see a confirmation dialog. (Proceeding will not affect this user's access to any other Cribl.Cloud Organizations they might own or be members of.)

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. At the top of your Cribl.Cloud portal, 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.

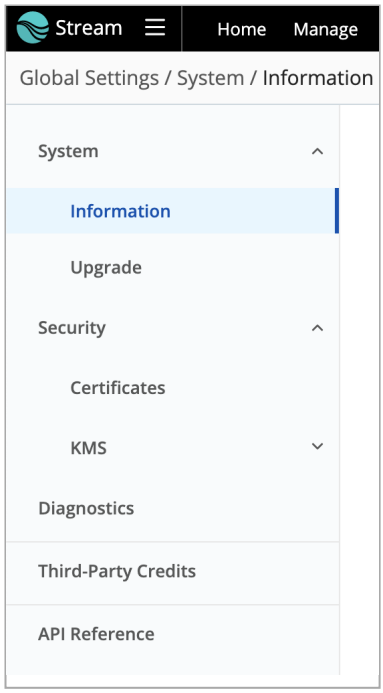
Differences from Self-Hosted Cribl Edge

A Cribl.Cloud deployment can differ from an on-prem/customer-managed Cribl Edge deployment in the following ways. Keep in mind all these differences as you navigate Cribl Edge's current UI, in-app help (including tooltips), and documentation.

Simplified Administration

Cribl.Cloud has been designed with options to accommodate everyone – from first-time evaluators, to Enterprise customers managing 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. Cribl manages many features on your behalf, allowing for a streamlined Settings left nav.



Below are the key options streamlined out of the free Cloud offering. Bear in mind that upgrading to an Enterprise plan will make many of these options configurable:

Simplified Distributed Architecture

Cribl.Cloud is preconfigured as a distributed deployment for Cribl Stream or Cribl Edge. With a Free or Standard plan, allows only a single Fleet.

Compared to self-hosted Cribl Edge, the **Settings** > **Worker Processes** and **Settings** > **Distributed Settings** links are omitted.

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 Settings** > **System** > **Git Settings** section is omitted. A local `git` client is preconfigured in your Cribl.Cloud portal. On Cribl.Cloud's top nav, use the **Global Config** link (branched icon) 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** and **Settings** > **Upgrade** links are omitted. Cribl handles restarts and version upgrades automatically on your behalf.

Simplified Access Management and Security

In Cribl.Cloud, you can manage access control for your Organization by clicking **Account** > **Organization** and selecting the **Members** tab. 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.

Transparent Licensing

The top nav's **Settings** > **Global Settings** > **Licensing** link is omitted. 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

Cribl is gradually narrowing the limitations listed in this section, as Cribl.Cloud gains feature parity with on-prem deployments:

- The Script Collector is available only on hybrid, customer-managed Workers. (This feature is currently not available on Cribl-managed Workers.)
- The System State Source is unavailable on Cribl-managed Workers. It is available on Cribl-managed Edge Nodes, however.
- The AppScope Source's **Filter Settings** are unavailable on Cribl-managed Workers. It is available on Cribl-managed Edge Nodes, however.
- The top nav's **Settings** > **Global Settings** > **Scripts** link is omitted from Cribl.Cloud, which currently does not support configuring or running shell scripts on hybrid or Cribl-managed Edge Nodes.
- The File System Collector and Filesystem Destination are available only on hybrid Edge Nodes.
- Persistent Queues can be configured on both hybrid and Cribl-managed Edge Nodes, with an Enterprise plan. On hybrid Edge Nodes, you can freely define the **Max queue size**, based on the disk space you provision. On Cribl-managed Edge Nodes, 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.)
- File-based Destinations support staging directories only on hybrid (not Cribl-managed) Edge Nodes.
- The Tee Function is available only on hybrid (not Cribl-managed) Edge Nodes.

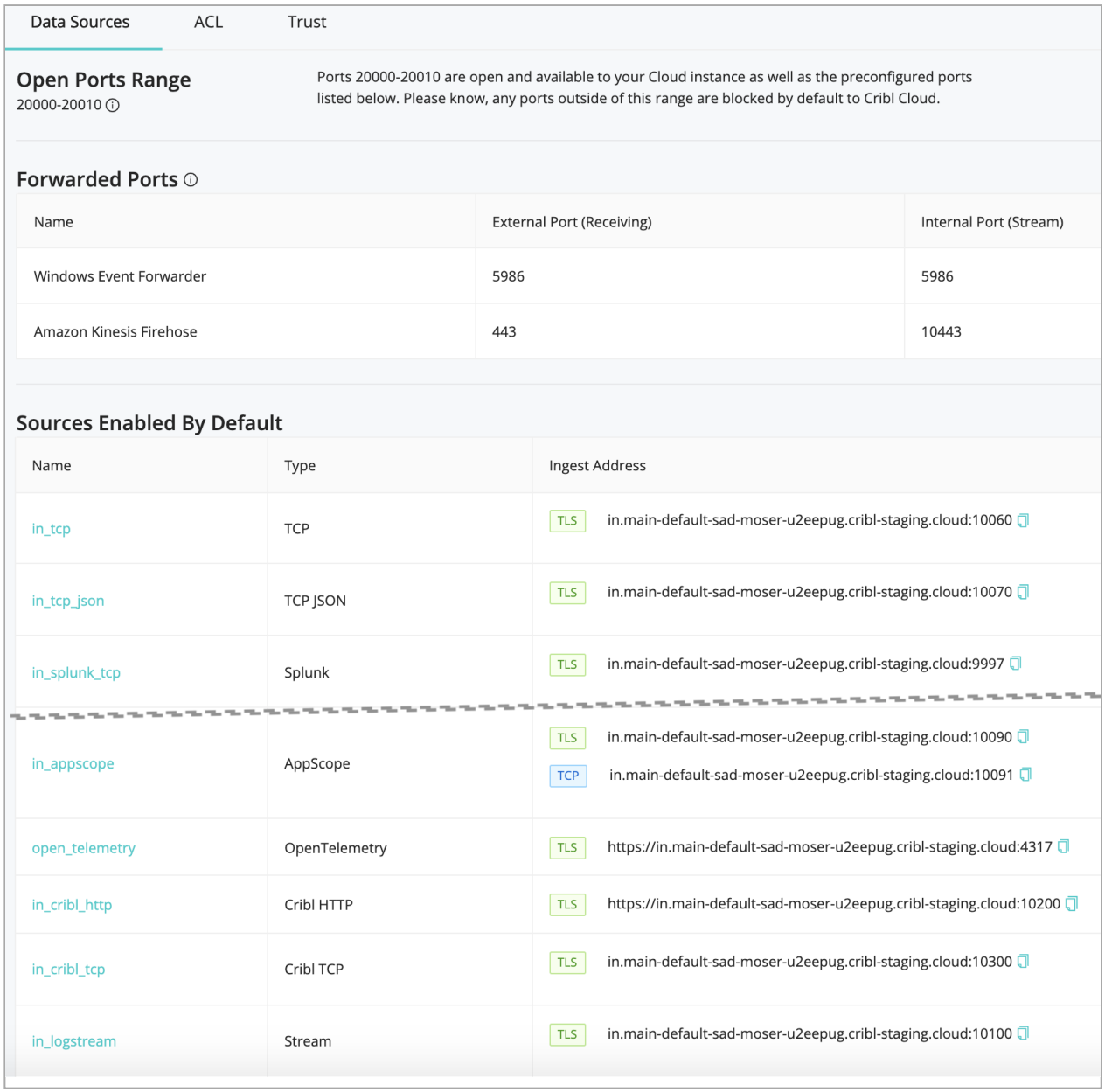
Support Options

At **Settings** > **Diagnostics**, you can generate diagnostic bundles and send them directly to Cribl Support. Currently, you cannot download diags. For all support options, see Working with Cribl Support.

Available Ports and TLS Configurations

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

The Cribl.Cloud portal's **Data Sources** tab displays the pre-enabled Sources, their endpoints, the reserved and available ports, and protocol details. For each existing Source listed here, Cribl recommends using the preconfigured endpoint and port to send data into Cribl Edge.



Available ports and TLS certificates

TLS Details

TLS encryption is pre-enabled for you on several Sources, also indicated on the Cribl.Cloud portal's **Data Sources** tab. All TLS is terminated by individual Nodes.

To enable TLS settings for additional Sources, use these configuration settings:

- **Private key path:** `/opt/criblcerts/criblcloud.key`

- **CA certificate path**: `/opt/criblcerts/criblcloud.crt`
- **Minimum TLS version**: `TLSv1.2`

Currently, Cribl.Cloud does not enable you to import your own certificates for mutual TLS authentication. Cribl.Cloud uses TLS to provide encryption in the wire, but leaves authentication at the protocol layer – e.g., Splunk HEC or S2S tokens, Kafka authorization, etc.

> **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, but is available on hybrid Workers.

The Cribl Internal Source's CriblLogs option is unavailable in Cribl-managed Stream instances, but it is available in Cribl Edge, and in hybrid Workers' Stream instances. The Cribl Internal > CriblMetrics option is available in all of the above combinations.

> In a preconfigured Source's configuration, never change the **Address** field, even though the UI shows an editable field. If you change these fields' value, the Source will not work as expected.
>
> After you create a Source and deploy the changes, it can take a few minutes for the Source to become available in Cribl.Cloud's load balancer. However, Cribl Edge will open the port, and will be able to receive data, immediately.

Enterprise Cloud

With a Cribl.Cloud Enterprise plan, you have the same options and flexibility as with an Enterprise license for an on-prem Cribl Edge distributed deployment – and more. (See Pricing for comparisons between Cloud plans and on-prem licenses.)

These options include configuring and managing multiple Worker Groups or Fleets, Notifications, Google SSO authentication, and Permission- or Role-based access control to Cribl Edge resources.

For other Enterprise features, see Pricing.

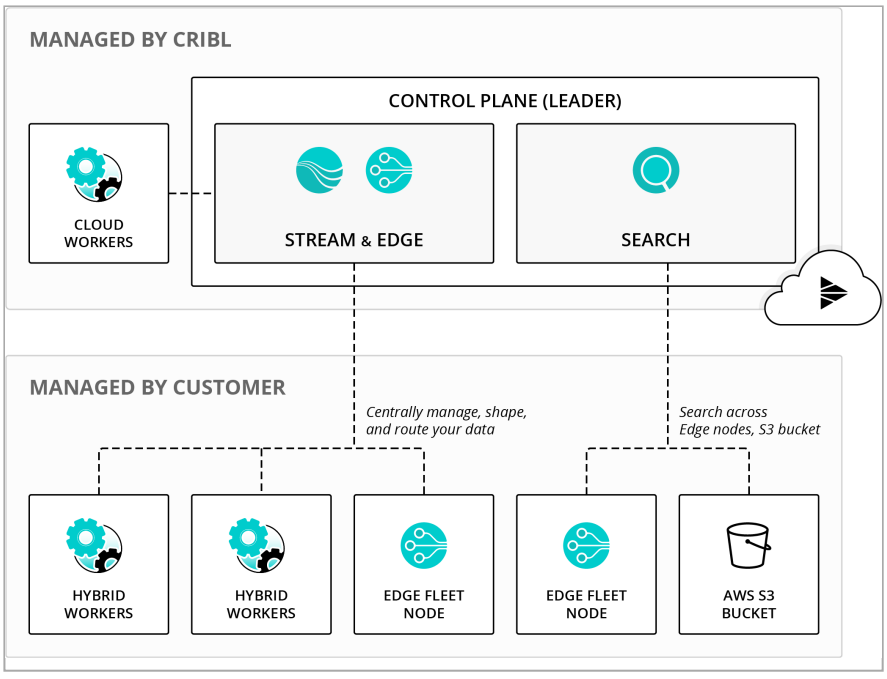
Cribl.Cloud Enterprise also adds:

- Full control of Member Permissions on your Cribl.Cloud Organization.
- The hybrid deployment option, described just below.
- The Leader resides in Cribl.Cloud, with access to diverse Worker deployments. Cribl manages the Leader's availability.

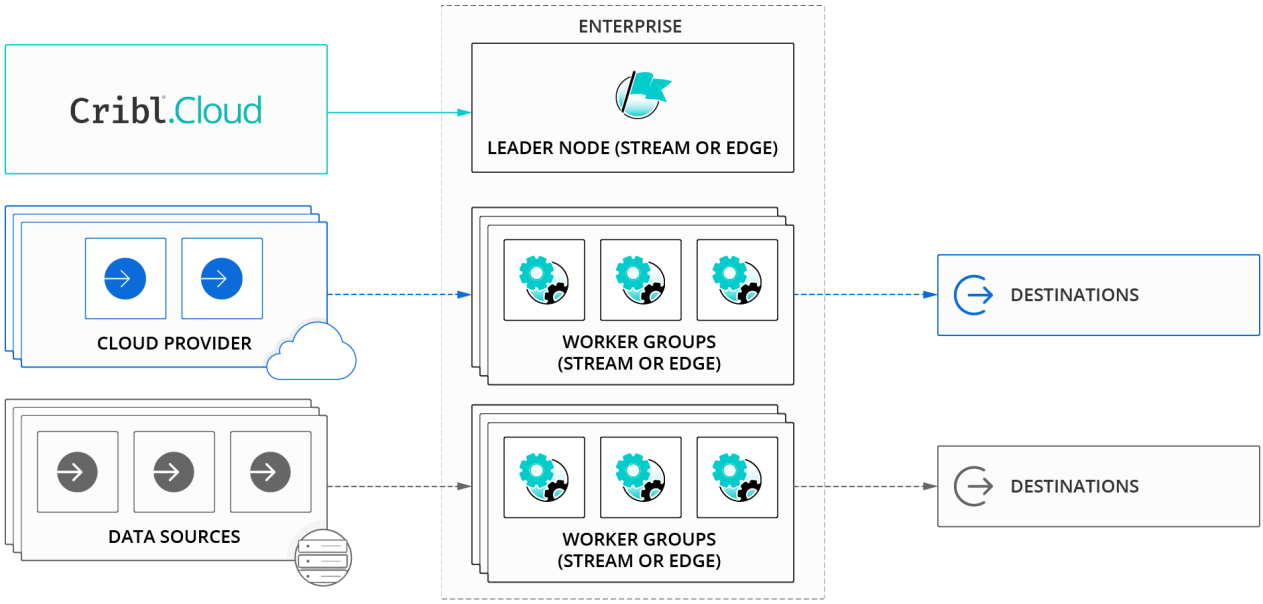
Hybrid Deployment

The diagrams below show the comparative flexibility of a Cribl.Cloud hybrid 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, 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 on port 443 to https://cdn.cribl.io.
- 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 Cloud hybrid 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/Cribl 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.
- Configure hybrid Workers by logging directly into their UI, then selecting **Settings** > **Global Settings** > **Distributed Settings**. Make sure the **Mode** is set to **Managed Worker** or **Managed Edge** (which might require a restart).
- Then select the **Leader Settings** left tab, and ensure a consistent entry in the **Address** field.
- In Cloud hybrid 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.

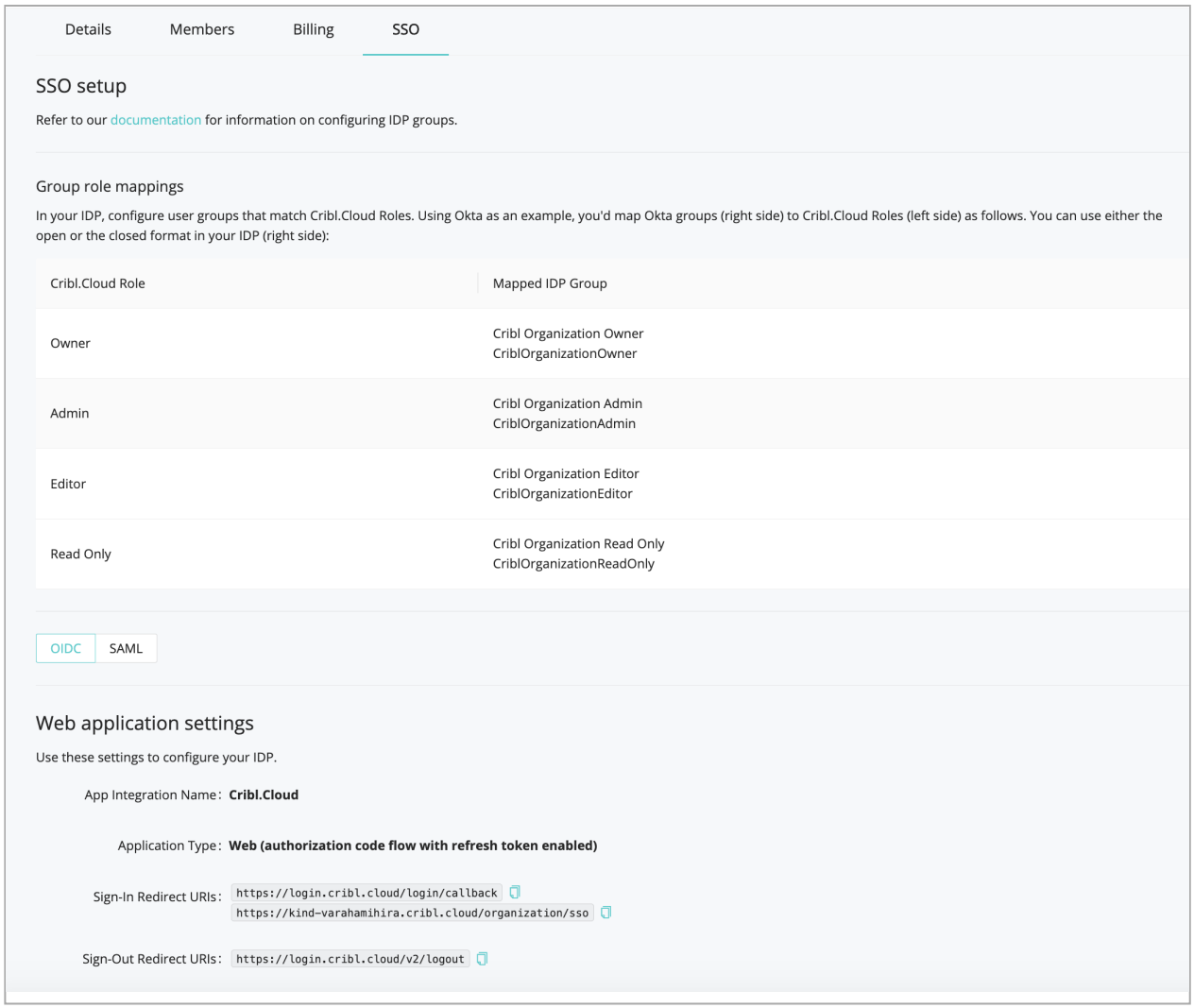
2.2. Cloud SSO Setup

2.2.1. Cribl.Cloud SSO Setup

The pages in this section outline how, with a Cribl.Cloud Enterprise plan, you can set up a Single Sign-On (SSO) integration between your identity provider and your Cribl.Cloud portal. The following pages cover both OIDC and SAML authentication options:

- Common SSO Setup Steps
- OIDC/Okta Setup Example
- SAML/Okta Setup Examples
- SAML/Azure AD Setup Examples
- Final SSO Steps & Troubleshooting

SSO integration requires you to perform certain configuration steps in your identity provider (IDP), and to then submit corresponding information to Cribl. As of Cribl Edge 4.0, you can submit these details directly on your Cribl.Cloud portal's Organization page.



Organization page SSO tab

This section covers both sides of the process. For additional details specifically about integrating Cribl Edge with Okta, see SSO/Okta Configuration.

The general steps to set up a Single Sign-On (SSO) integration between your identity provider and your Cribl.Cloud portal are:

1. Invite at least one SSO admin to your Cribl.Cloud Organization from a fallback, separate email domain.
2. In your identity provider (IDP), configure user groups that map to Cribl.Cloud's four predefined Roles.
3. 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.

4. Submit your app's configuration details to Cribl on your Cribl.Cloud portal's **Organization** page > **SSO** tab. (This will complete your SSO setup on the Cribl side.)

5. In your IDP, assign groups to your users, matching the Role that each group of users should have in Cribl.Cloud.

6. In your IDP, assign the OIDC/SAML app to the Organization's owner, and to other Cribl.Cloud users.

2.2.2. Common SSO Setup Steps

Whether you're configuring Single Sign-On (SSO) via OIDC or SAML, these initial preparation steps are the same.

Set Up Fallback Access

From your Cribl.Cloud Organization, send at least one administrator of your IDP a duplicate member invitation at an email domain that's separate from the corporate domain on which you're configuring SSO. This will ensure backup access if SSO configuration breaks.

Configure Groups

In your IDP, configure user groups that match Cribl.Cloud Roles. Using Okta as an example, you'd map Okta groups (right side) to Cribl.Cloud Roles (left side) as follows.

You can use either the open or the closed format in your IDP (right side). You can also add prefixes or suffixes, as outlined below in Group Configuration Options.

CRIBL.CLOUD ROLE	IDP GROUP NAME
Owner	Cribl Organization Owner /or/ CriblOrganizationOwner
Admin	Cribl Organization Admin /or/ CriblOrganizationAdmin
Editor	Cribl Organization Editor /or/ CriblOrganizationEditor
Read-Only	Cribl Organization Read Only /or/ CriblOrganizationReadOnly

Group Configuration Best Practices

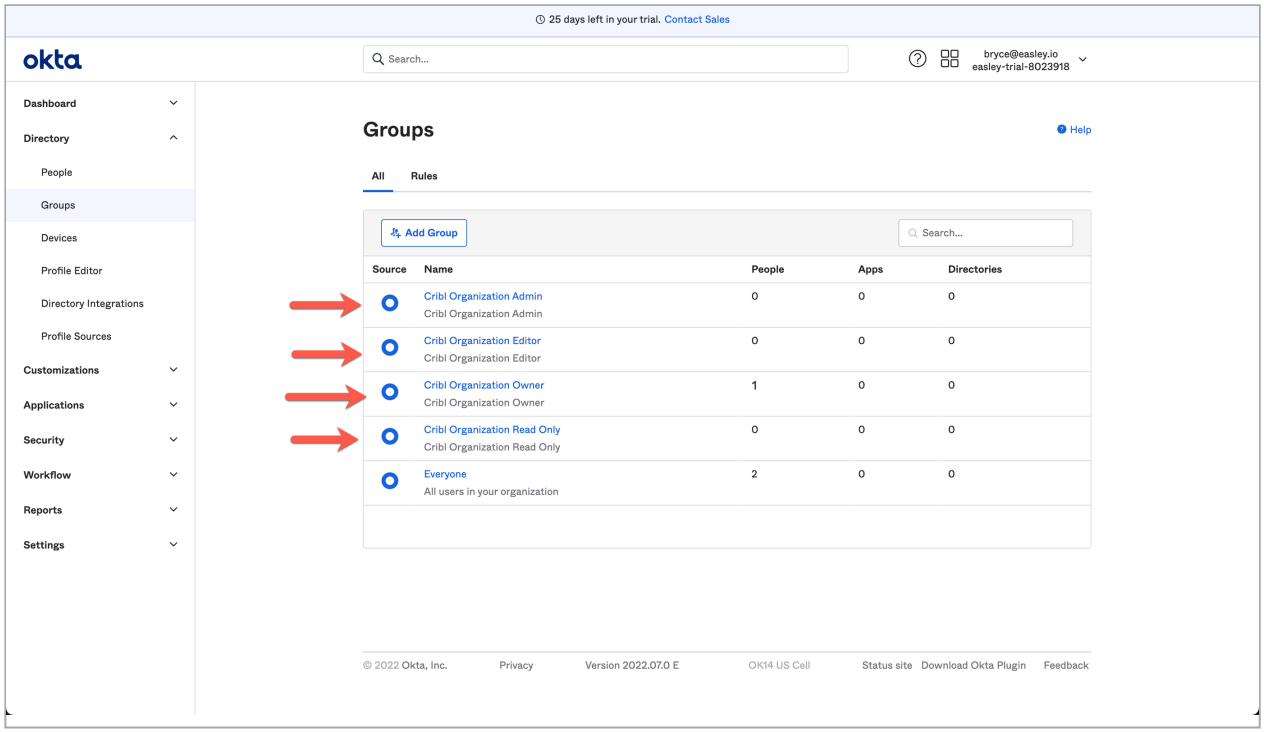
- A Cribl.Cloud Organization's Owner Role can be shared and transferred among multiple users. This facilitates gradual ownership transfers, corporate reorganizations, and other scenarios.
- Those users should be in both the Owner and Admin groups in your IDP.
  (This enables them to acquire all needed 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.)

Group Configuration Options

You can freely add prefixes or suffixes to Group Names that follow the above examples' formats. Cribl will ignore these additions when mapping IDP groups to Cribl Roles. Examples:

- `APP-SIEM-203916-ISRM-CriblOrganizationOwner`
- `CriblOrganizationEditor-420`

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

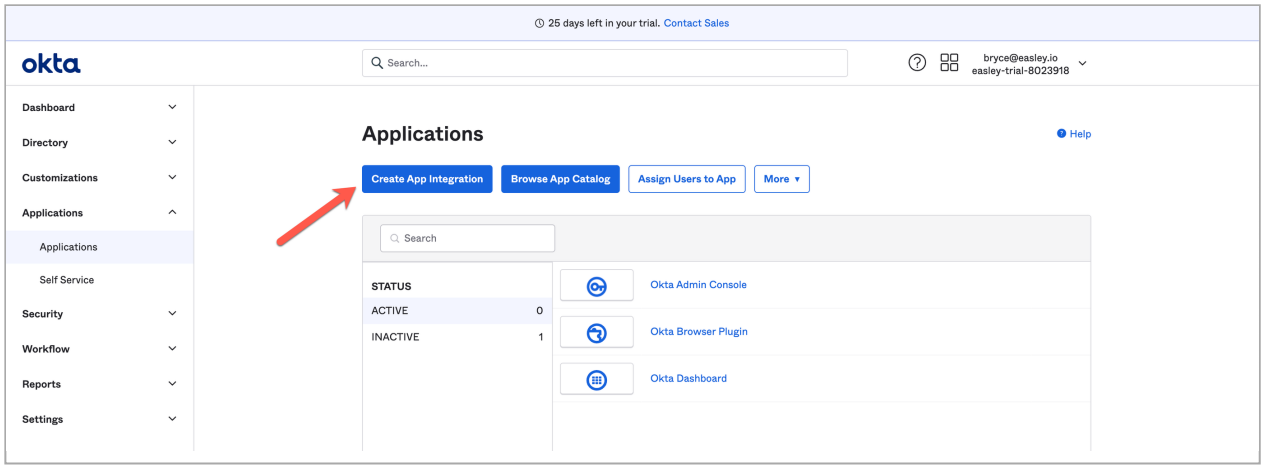
2.2.3. OIDC/Okta Setup Example

This page expands on the overview for OIDC, offering a detailed walkthrough with Okta as the example IDP.

> Cribl.Cloud supports only OIDC backchannel authentication, not front-channel.

Create OIDC App Integration

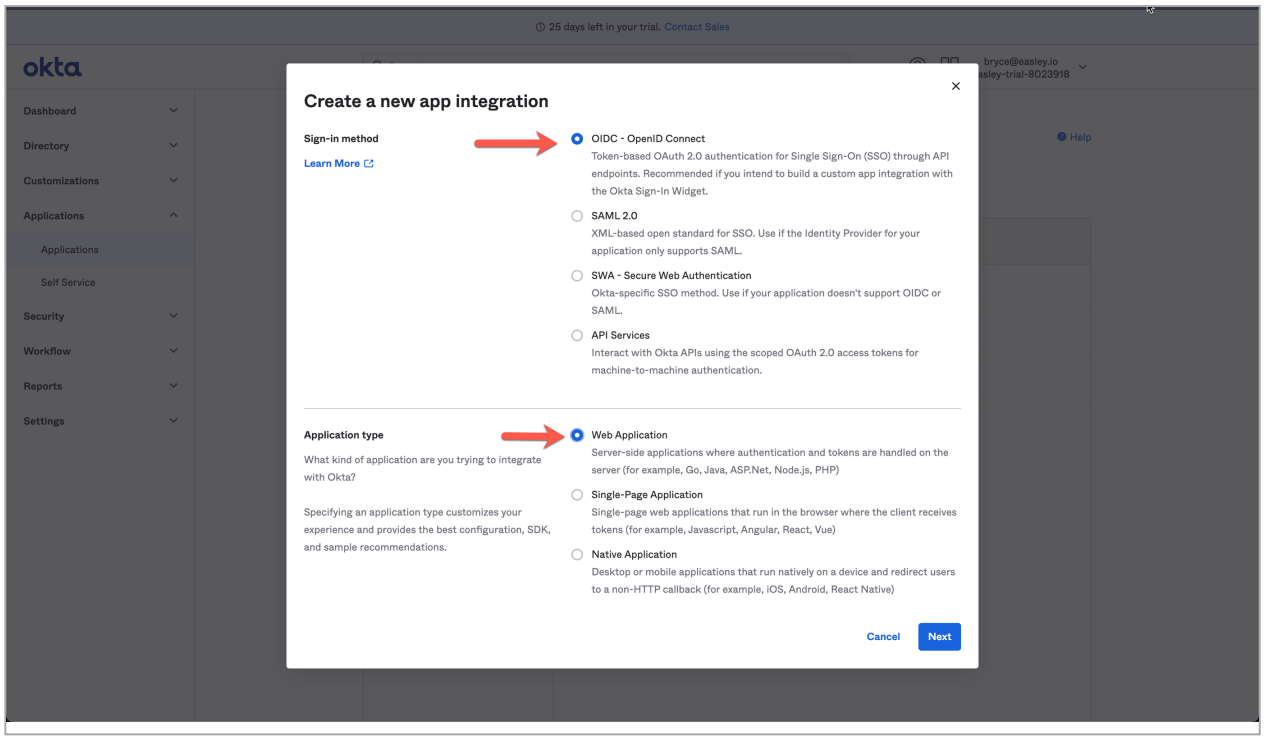
To create your app integration within Okta, start here:



Creating an app integration

Next, configure the app integration with the options below.

- **Sign-in method**: `OIDC - OpenID Connect`
- **Application type**: `Web Application`

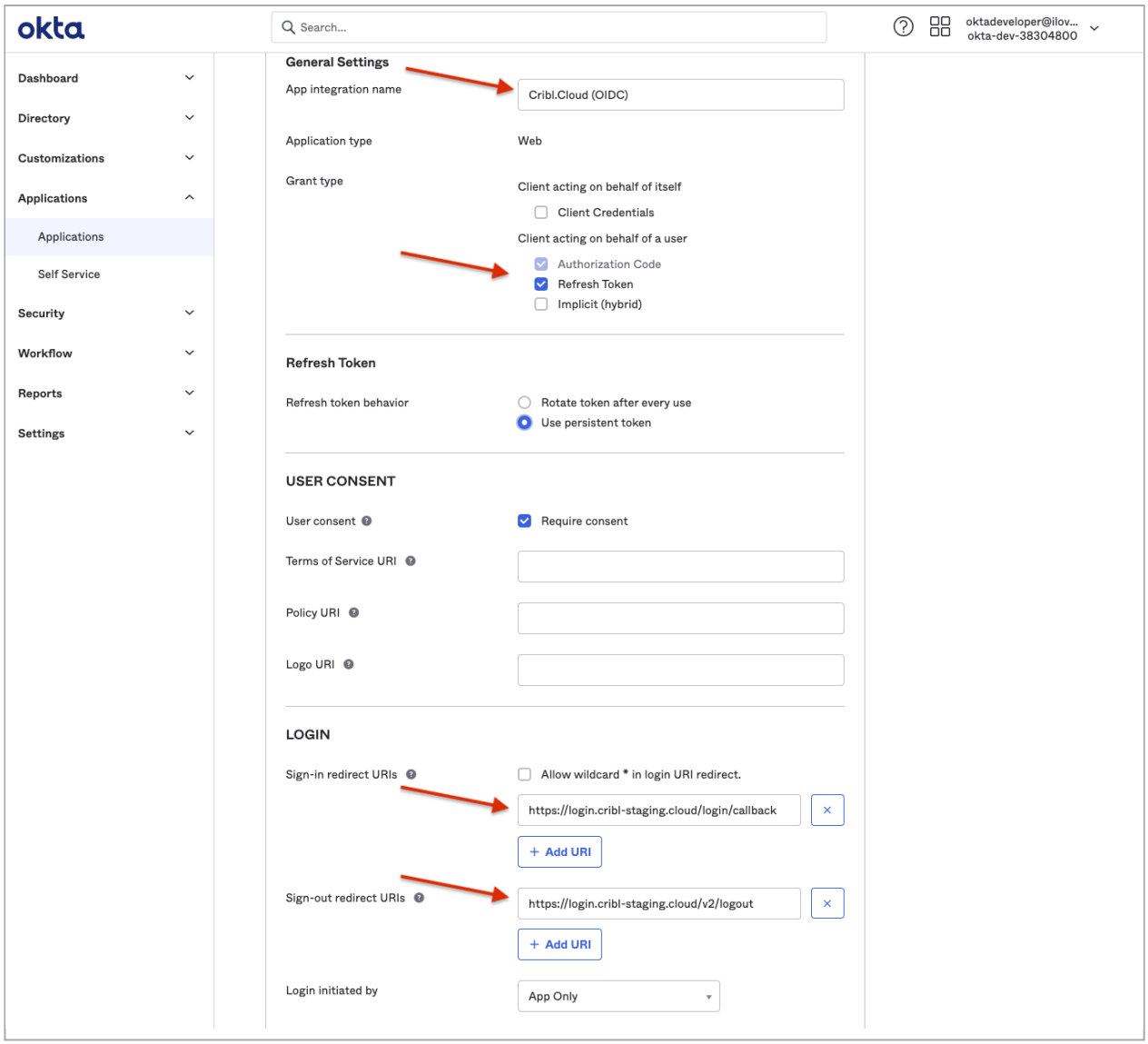


Configuring new Okta app integration

General Settings

Configure the app integration's **General Settings** with the options below.

- **App Integration Name**: `Cribl.Cloud`
- **Sign-In Redirect URIs**: https://login.cribl.cloud/login/callback
- **Sign-Out Redirect URIs**: https://login.cribl.cloud/v2/logout
- **Grant Type**: `Authorization Code, Refresh Token`
- **Scopes**: `openid profile email groups`



App integration name and URIs

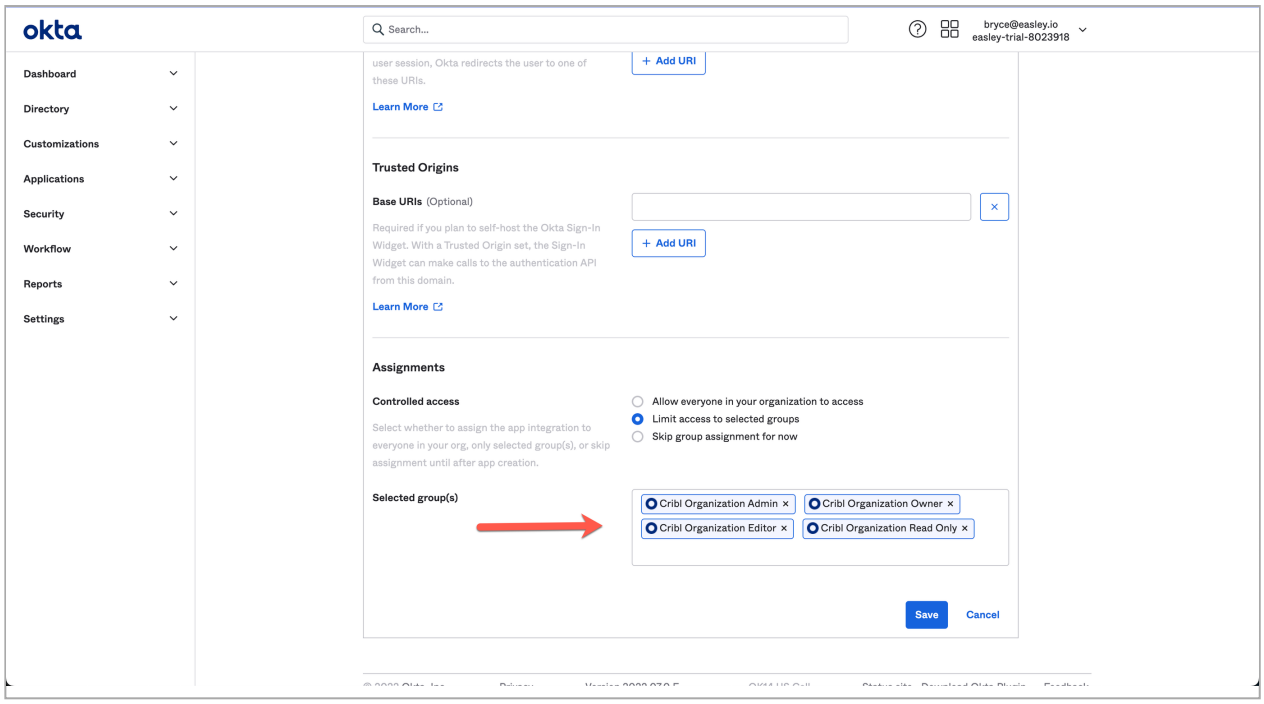
If your IDP is PingOne, you must also configure this (non-Okta) option:

- **Authentication Options**: `Allow Client Secret`

Assignments

Configure the **Assignments** pane with the options below.

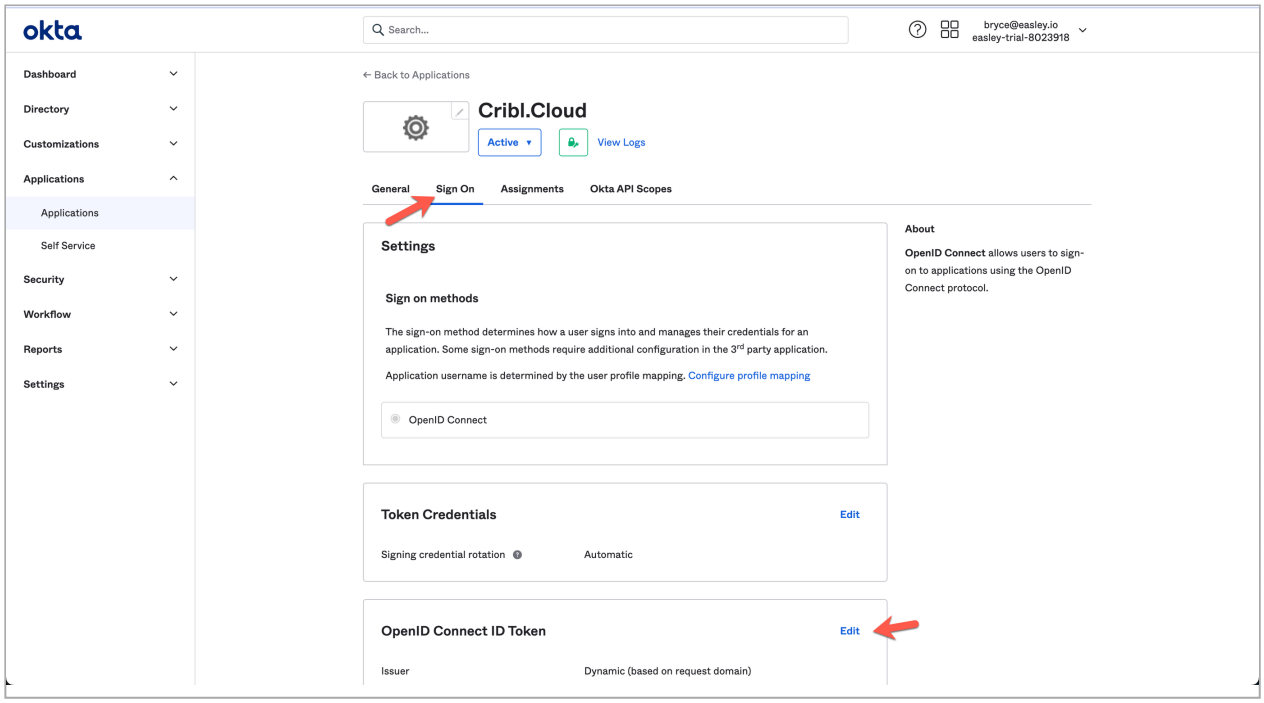
- **Controlled access**: `Limited access to selected groups`
- **Selected groups**: <The groups you mapped in Configuring Groups>



Assignments configuration

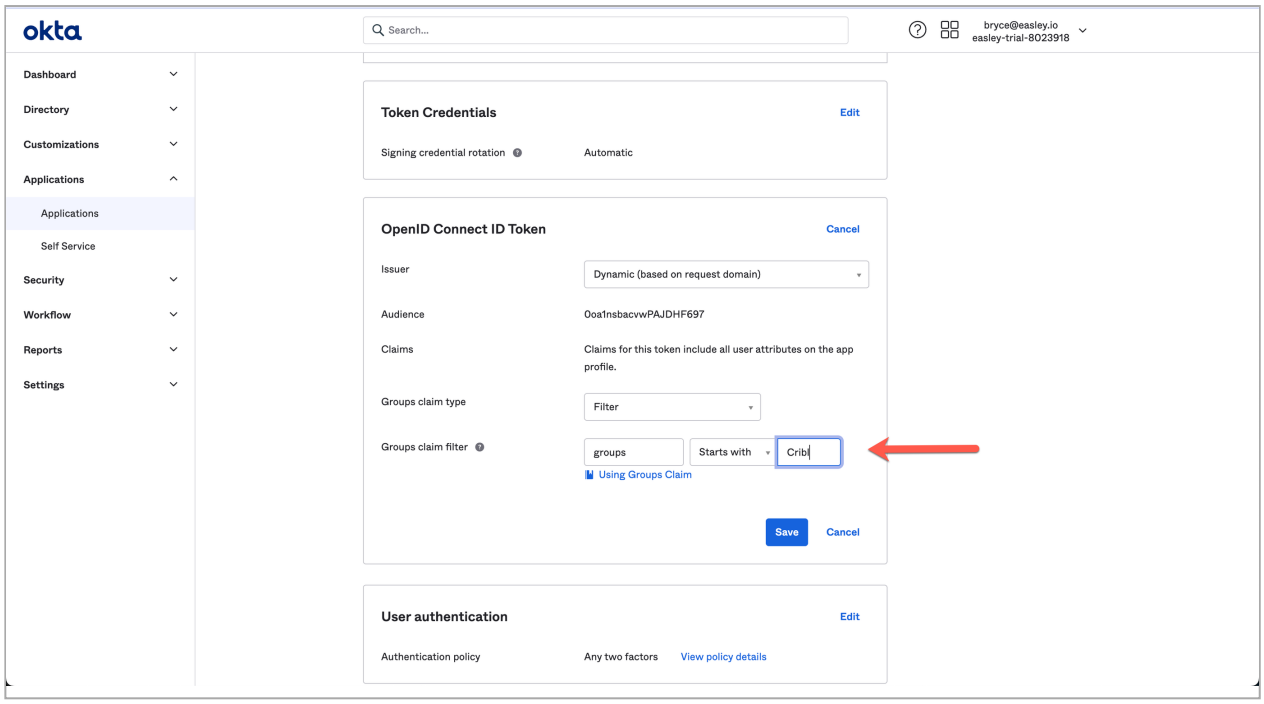
Sign-On Tab

Edit the **OpenID Connect ID Token** section, as shown here:



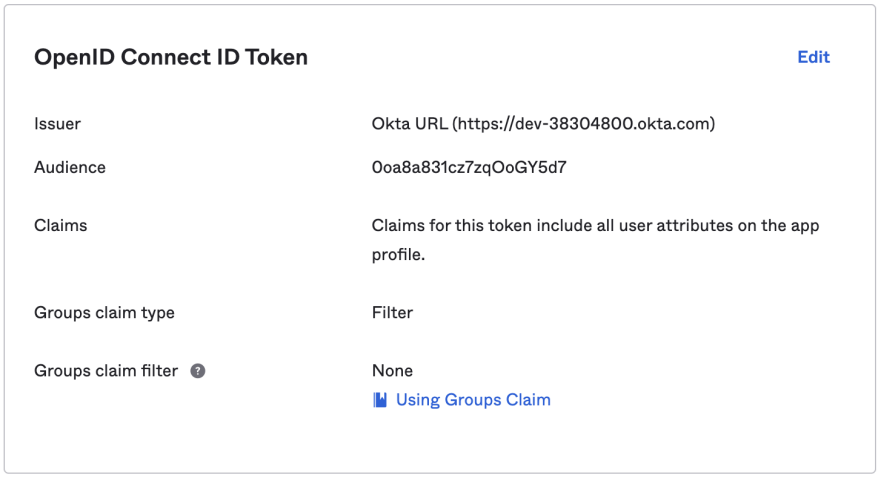
Sign-On tab's Settings/Tokens sections

Set the **Groups claim filter** to: `groups : Starts with : Cribl`, as shown below:



OIDC token: configuring Groups claim filter

Change the **Issuer** from `Dynamic` to `Okta URL`, to get the **Issuer URL** to provide to Cribl in the next section:

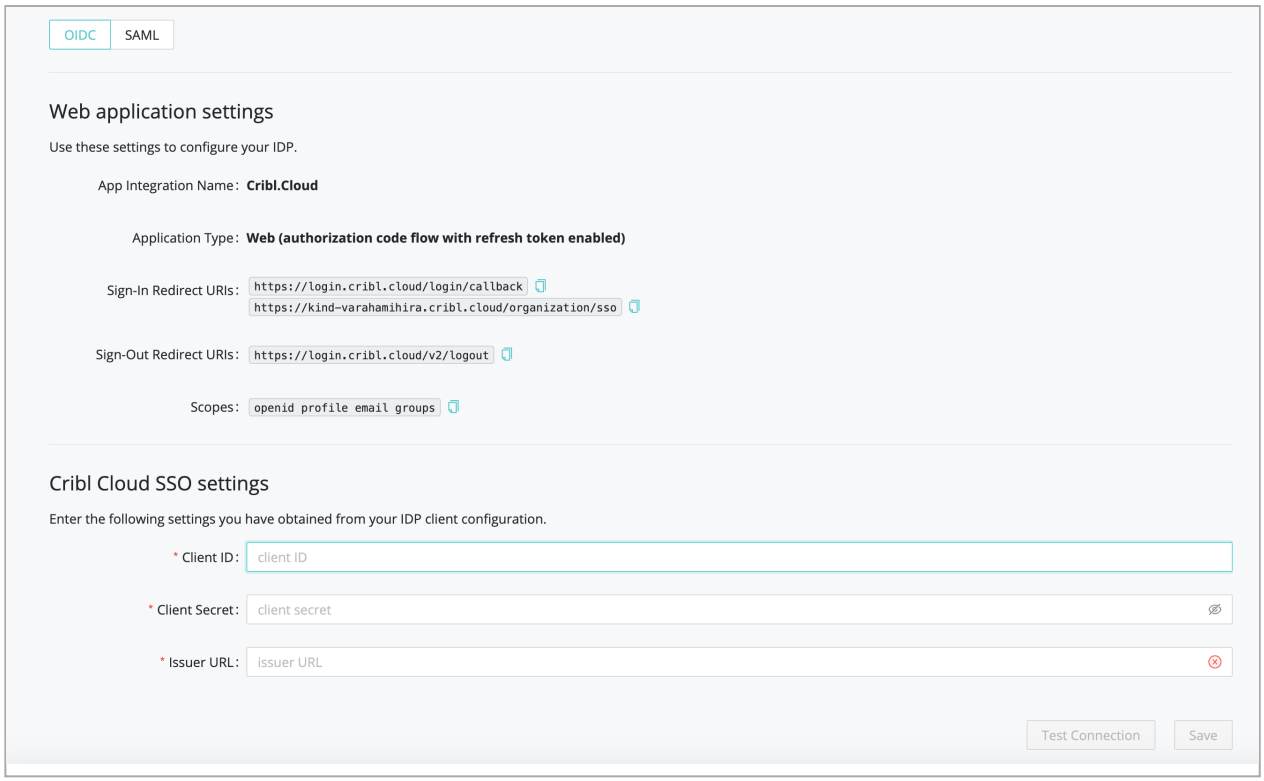


OIDC token: obtaining Issuer URL

Submit Your App Info to Cribl

This concludes the setup you need to do in Okta (or other IDP). Next, provide Cribl essential details about your application, to implement SSO setup on the Cribl side.

On your Cribl.Cloud portal's Organization page > **SSO** tab, select the **OIDC** lower tab.

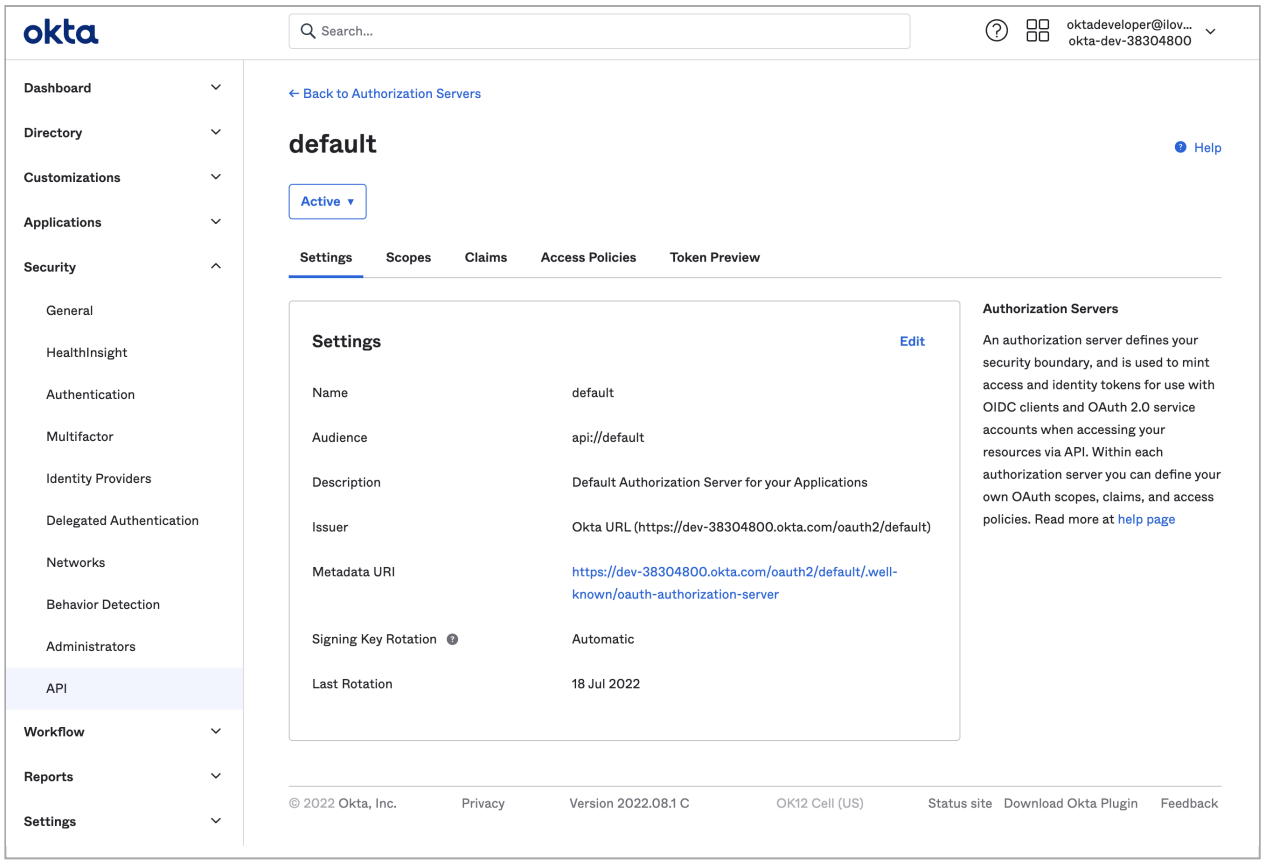


SSO > OIDC lower tab

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

Continuing with our Okta example, here's where you'd copy the Issuer URL from your Okta app:

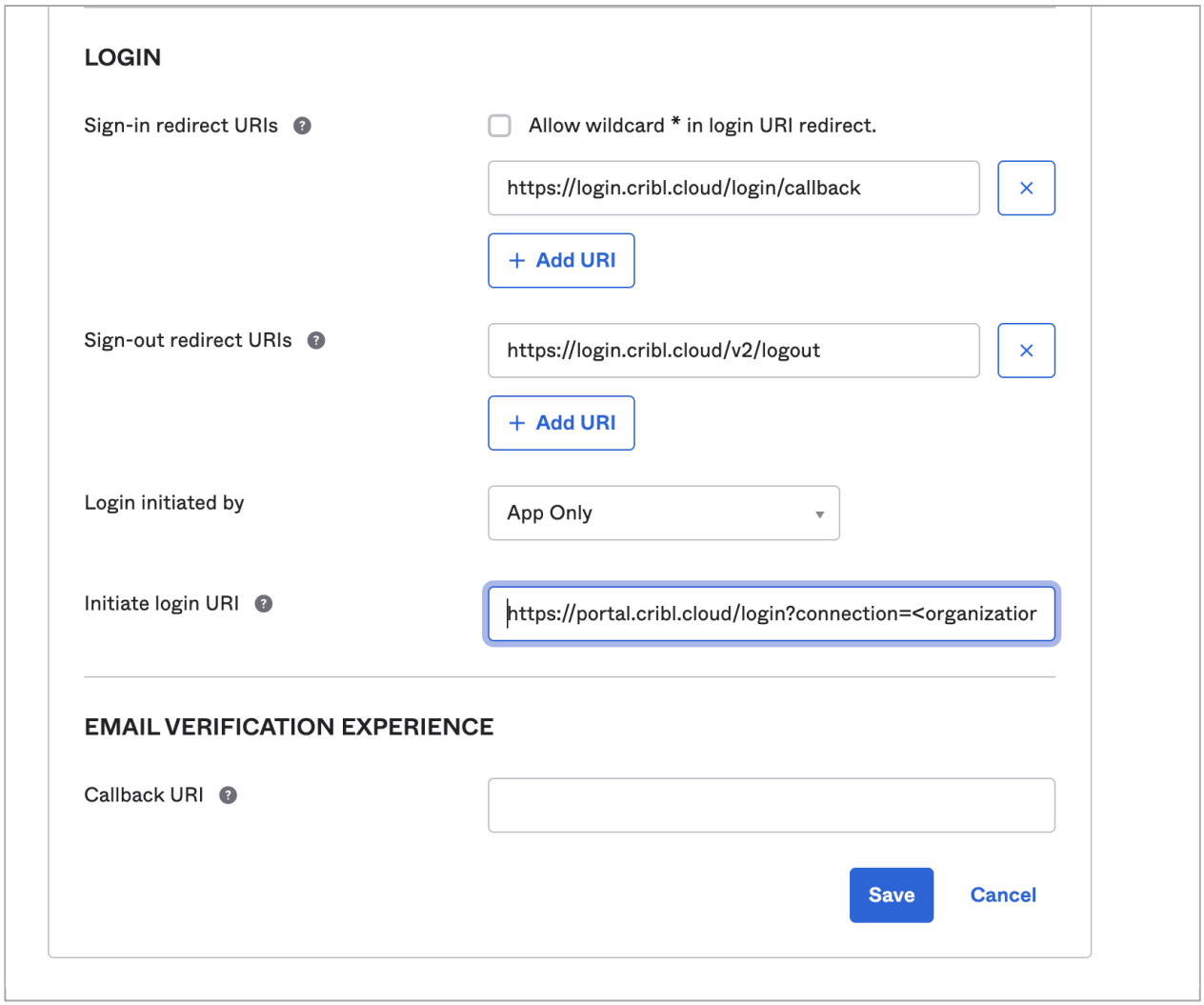


Obtaining Issuer URL from Okta

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. Click that application, and select **General Settings** > **Edit**.
4. In the **Initiate login URI** field, enter `https://portal.cribl.cloud/login?connection=<organizationID>` (where `<organizationID>` is your Cribl.Cloud Organization's ID).



Adding the Initiate Login URI value

5. Click **Save** to complete the chiclet.

Link Existing Users

To ensure that your Cribl.Cloud Organization's local users have a smooth transition to SSO, see Final SSO Steps & Troubleshooting.

2.2.4. SAML/Azure AD Setup Examples

This example uses Azure Active Directory as the identity provider (IDP).

Get URL and ID from Cribl

Cribl's terminology corresponds to Azure AD's terminology as follows:

CRIBL.CLOUD	AZURE AD
Single Sign-On URL	Reply URL (Assertion Consumer Service URL)
Audience URI	Identifier (Entity ID)

Create an Enterprise Application

In Azure Active Directory:

1. Select **Enterprise applications** (on the left) > **New application** > **Create your own 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. Click **Create**.

Assign Groups

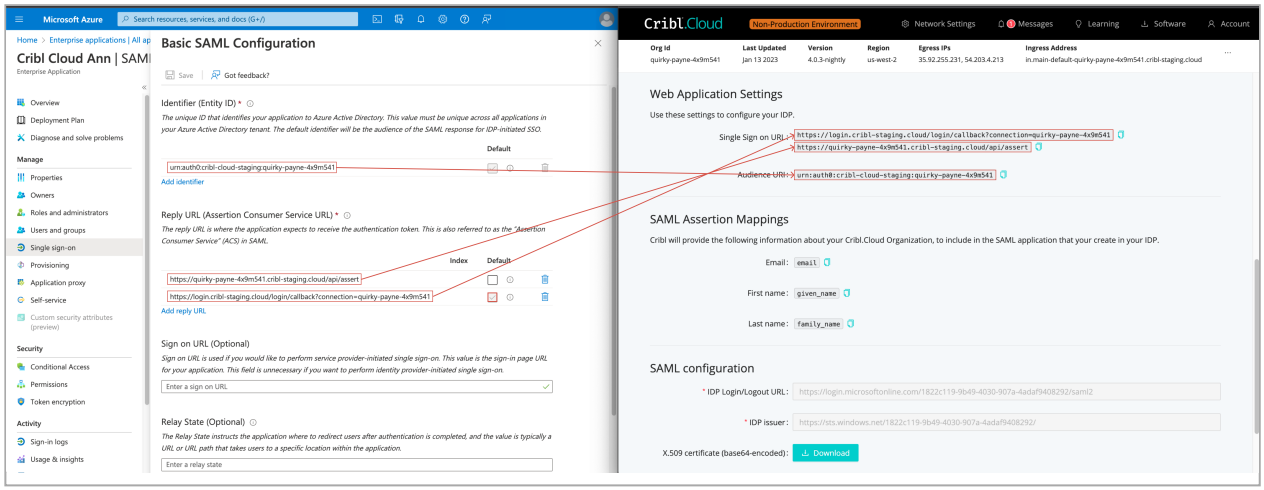
From Azure AD's left nav:

1. Select **Users and groups**.
2. Select **Add user/group**.
3. Add the Cribl groups you created in Configure Groups.
4. Click **Assign** after selecting Groups.

Configure Single Sign-On

From Azure AD's left nav, select **Single sign-on** > **SAML** to open the **Basic SAML Configuration** page. Then, as shown in the screenshot below:

1. Select **Add identifier** and enter the **Audience URI** value from Cribl.Cloud's SAML setup page.
2. 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 check the checkbox to make it the **Default**.



Configuring SSO in Azure AD

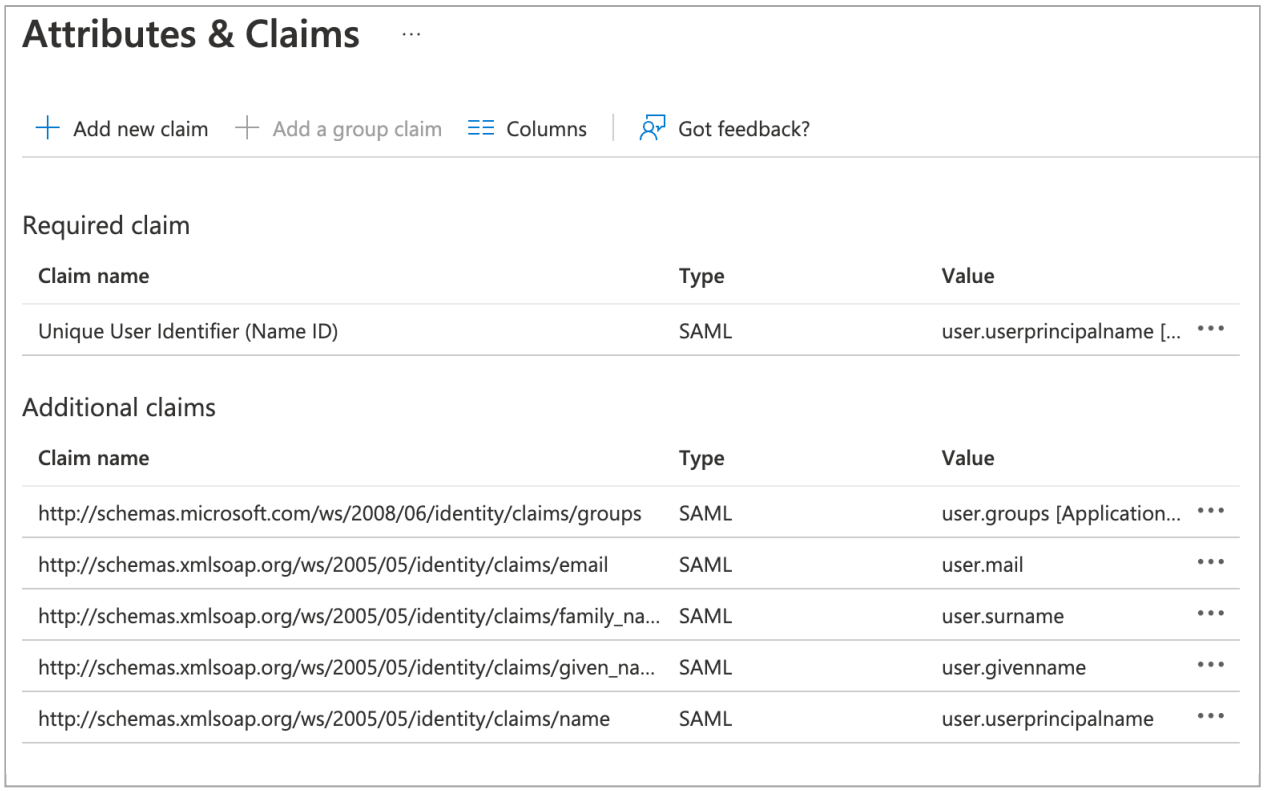
Configure Attributes and Groups Claims

In Azure AD, edit **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. As the **Source Attribute**, select: `Cloud-only group display names (Preview)`.
3. Accept the defaults for everything else, and save the new settings.



SAML Azure AD Attribute

Submit Your App Info to Cribl

After you've created the SAML app integration in your IDP, provide Cribl essential metadata about your application, to implement SSO setup on the Cribl side.

1. On your Cribl.Cloud portal's Organization page > **SSO** tab, select the **SAML** lower tab.
2. Set the **IDP Login/Logout URL** to your Azure AD's **Set up CloudSAML** section > **Login URL** value.
3. Set the **IDP issuer** to your Azure AD's **Set up CloudSAML** section > **Azure AD Identifier** value.
4. To set the **X.509 certificate (base64-encoded)**, navigate to Azure AD's **SAML Certificates** section and download your **Base64 Certificate**.
5. Click **Test Connection**.
6. When you've verified the connection, click **Save** to complete your submission.

Link Existing Users

To ensure that your Cribl.Cloud Organization's local users have a smooth transition to SSO, see Final SSO Steps & Troubleshooting.

2.2.5. SAML/Okta Setup Examples

This example uses Okta as the identity provider (IDP).

Get URL and ID from Cribl

Cribl will provide the following information about your Cribl.Cloud Organization, to include in the SAML application that you create in your IDP:

1. Assertion Consumer Service URL.

   Okta calls this a "Single sign-on URL," and this is the first of two URLs that your Cribl Organization's **SSO** > **SAML** tab lists under the same label. Example: `https://login.cribl.cloud/login/callback?connection=<$organizationID>`

2. Test SSO URL.

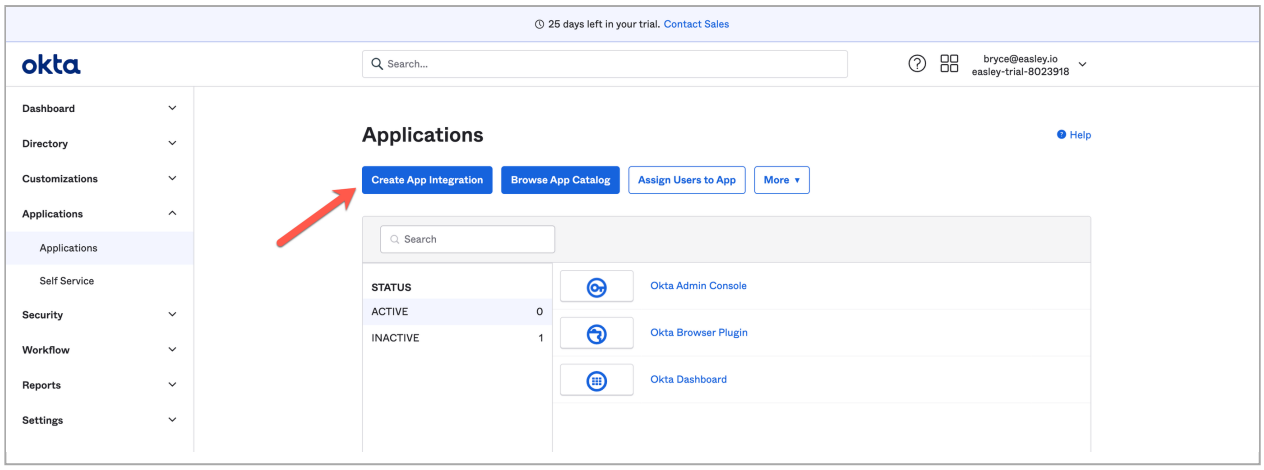
   Required to test your connection. Okta accepts this under "Other Requestable SSO URLs," and this is the second URL that your **SSO** > **SAML** tab lists under "Single sign-on URL." Example: `https://<$organizationID>.cribl.cloud/api/assert`

3. Entity ID.

   Okta calls this an "Audience URI (SP Entity ID)," and your **SSO** > **SAML** tab calls it just an "Audience URI." Example: `urn:auth0:cribl-cloud-prod:<$organizationID>`

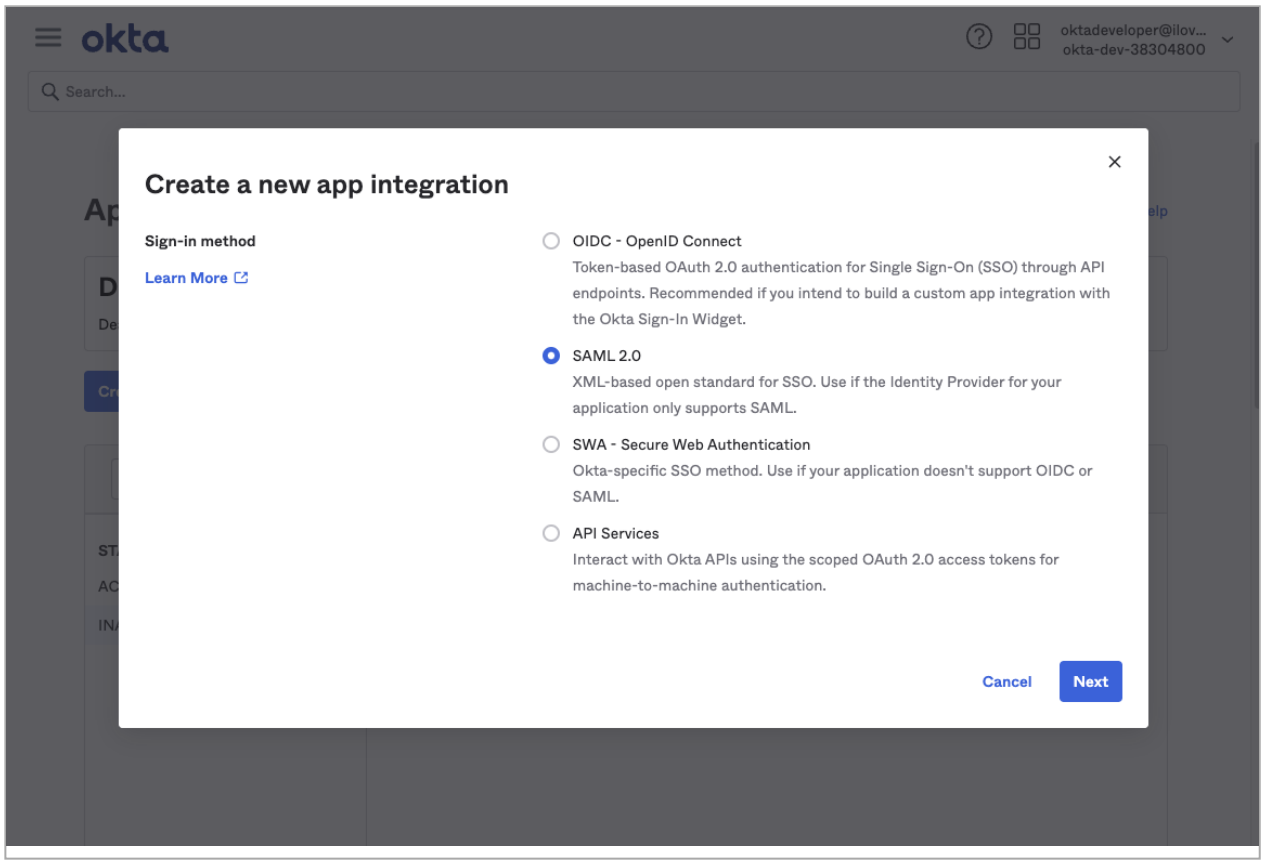
Create SAML 2.0 App Integration

Using Okta as our example, create a SAML app integration. Start here:



Creating an app integration

Next, create the app integration with **Sign-in method**: `SAML 2.0`.



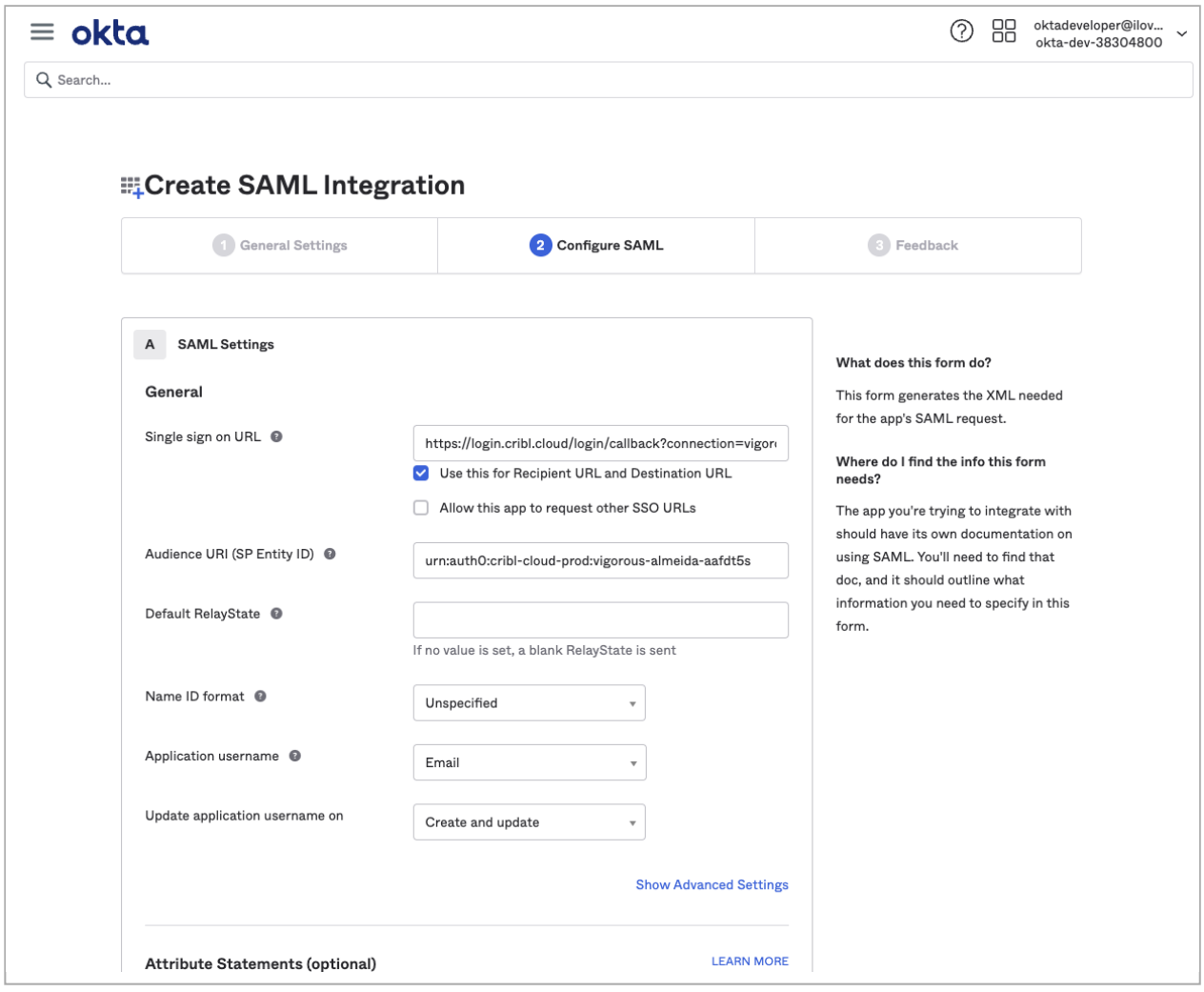
Specifying SAML 2.0 sign-in

Configure SAML Settings

In the app integration **SAML Settings** section, configure the following options.

- **Single sign-on URL (Assertion Consumer Service URL)**: `https://login.cribl.cloud/login/callback?connection=<$organizationID>`
- **Audience URI (SP Entity ID)**: `urn:auth0:cribl-cloud-prod:<$organizationID>`
- **Application username**: `Email`

> The `nameidentifier` assertion in SAML responses must be the user's `Email`.

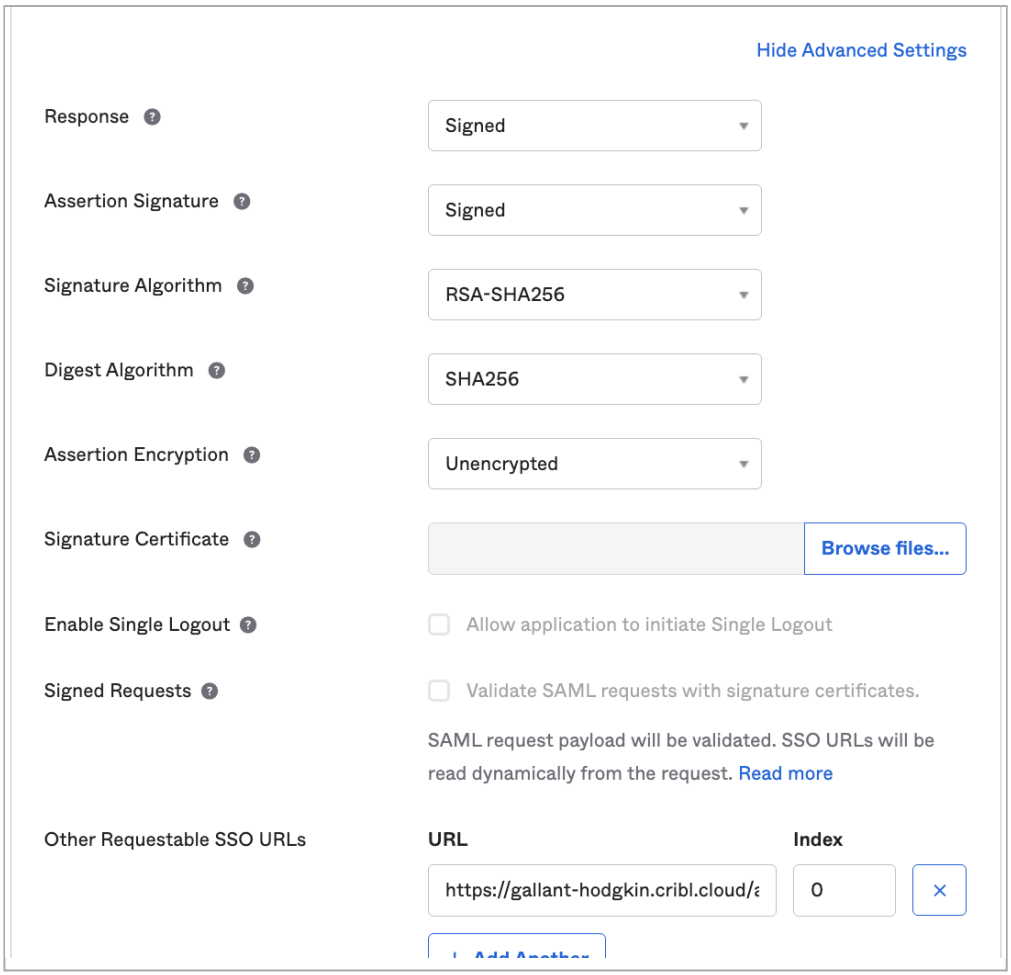


Configuring SAML Settings

Next, click the **SAML Settings** section's **Show Advanced Settings** link. Then navigate down to configure a single row of **Other Requestable SSO URLs**, as follows:

- **URL**: From your Cribl.Cloud Organization's **SSO** > **SAML** tab, this is the second **Single sign-on URL**. It will be in this format: `https://<$organizationID>.cribl.cloud/api/assert`

- **Index**: Set this to `0`.

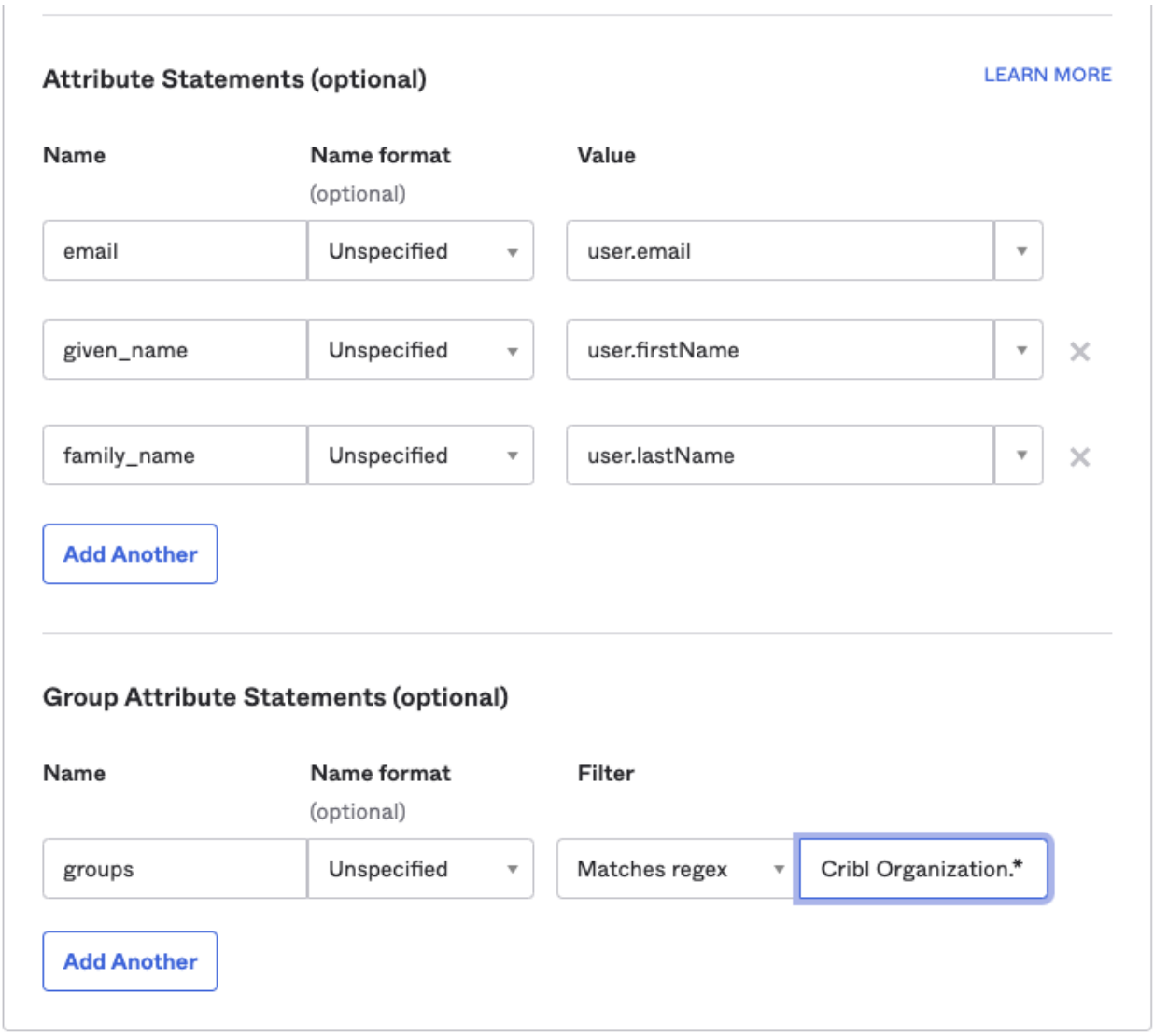


Configuring SAML Advanced Settings

Configure Attribute Statements

Configure **Attribute Statements** for these attributes, as shown below. Then save your app integration:

- `email`
- `given_name`
- `family_name`
- `groups`

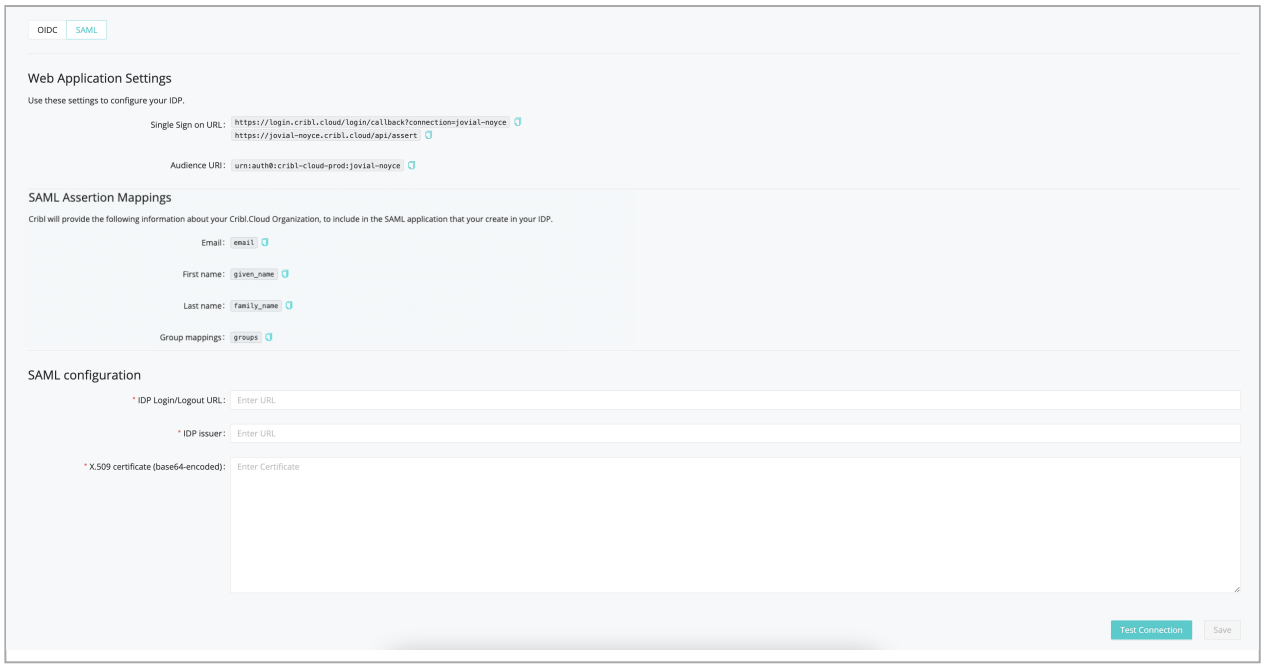


Configuring SAML Attribute Statements in Okta

Submit Your App Info to Cribl

After you've created the SAML app integration in your IDP, provide Cribl essential metadata about your application, to implement SSO setup on the Cribl side.

On your Cribl.Cloud portal's Organization page > **SSO** tab, select the **SAML** lower tab.

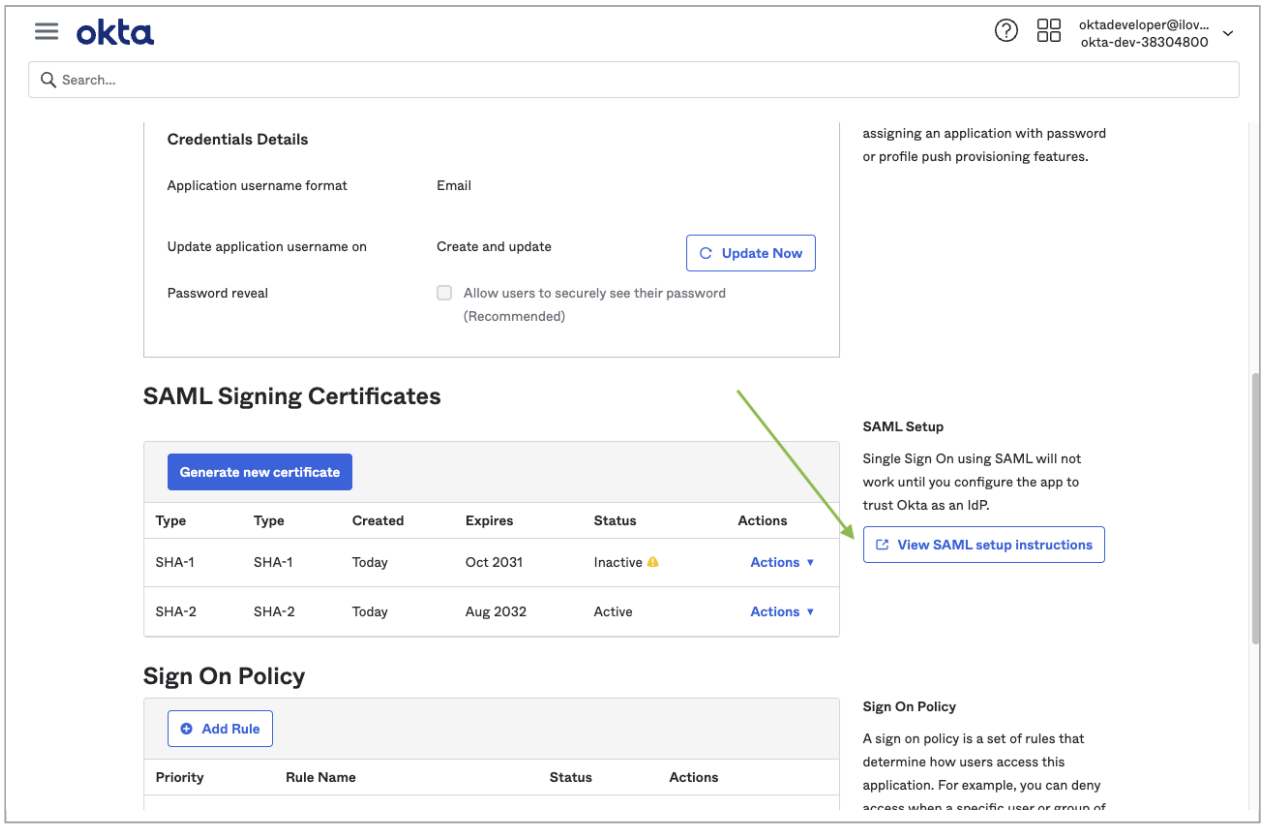


SSO > SAML lower tab

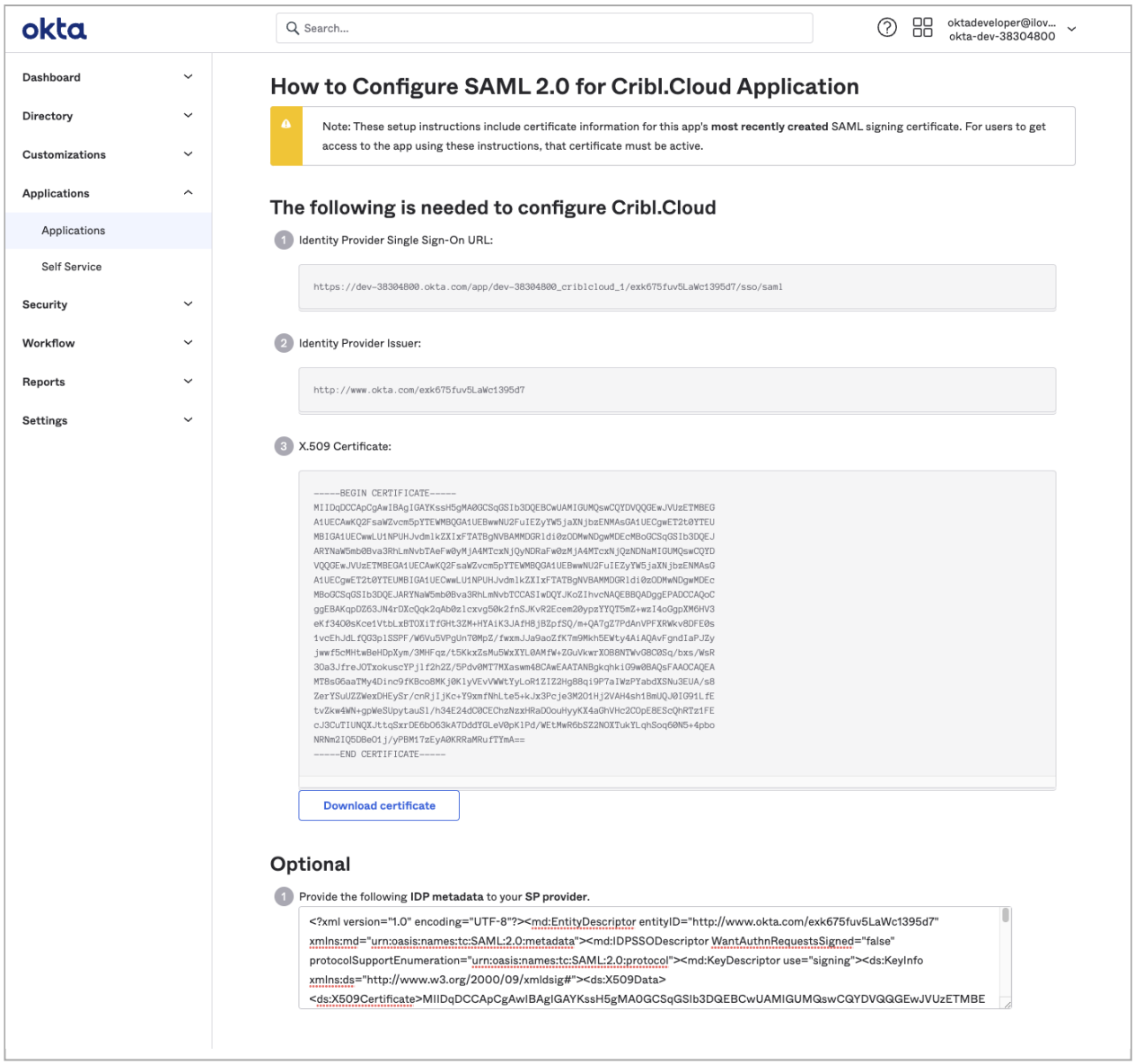
The **Web Application Settings** will be prefilled for you, and Cribl will also prefill the **SAML Assertion Mappings** based on the information you've registered with Cribl. So you only need to fill in the **SAML Configuration** section with the following details from your IDP client configuration:

- IDP SSO
- IDP Issuer
- X.509 Certificate

Continuing with our Okta example, these screenshots show how you'd export these metadata details from your Okta app:



Exporting SAML metadata



SAML metadata in Okta's UI

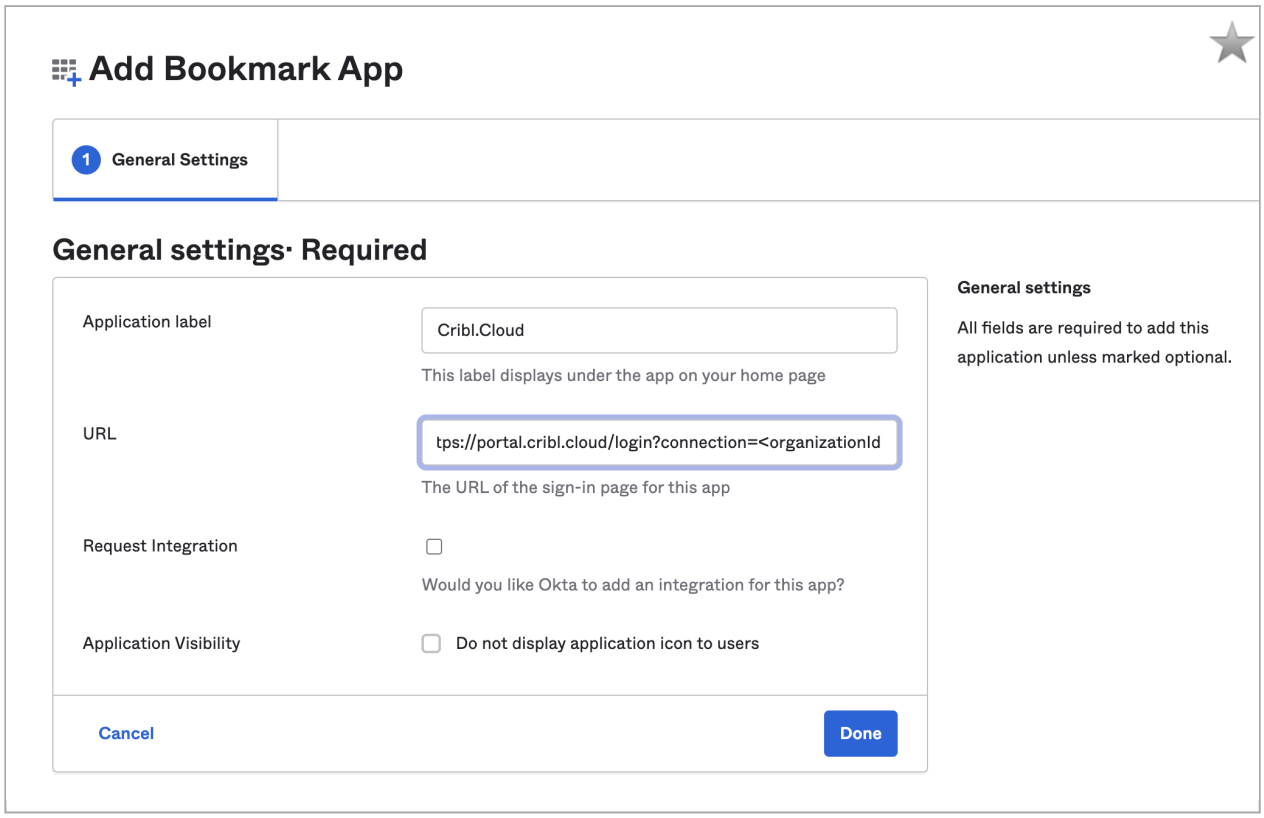
SAML/Okta Chiclet Setup (Optional)

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. Click **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, click **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.)

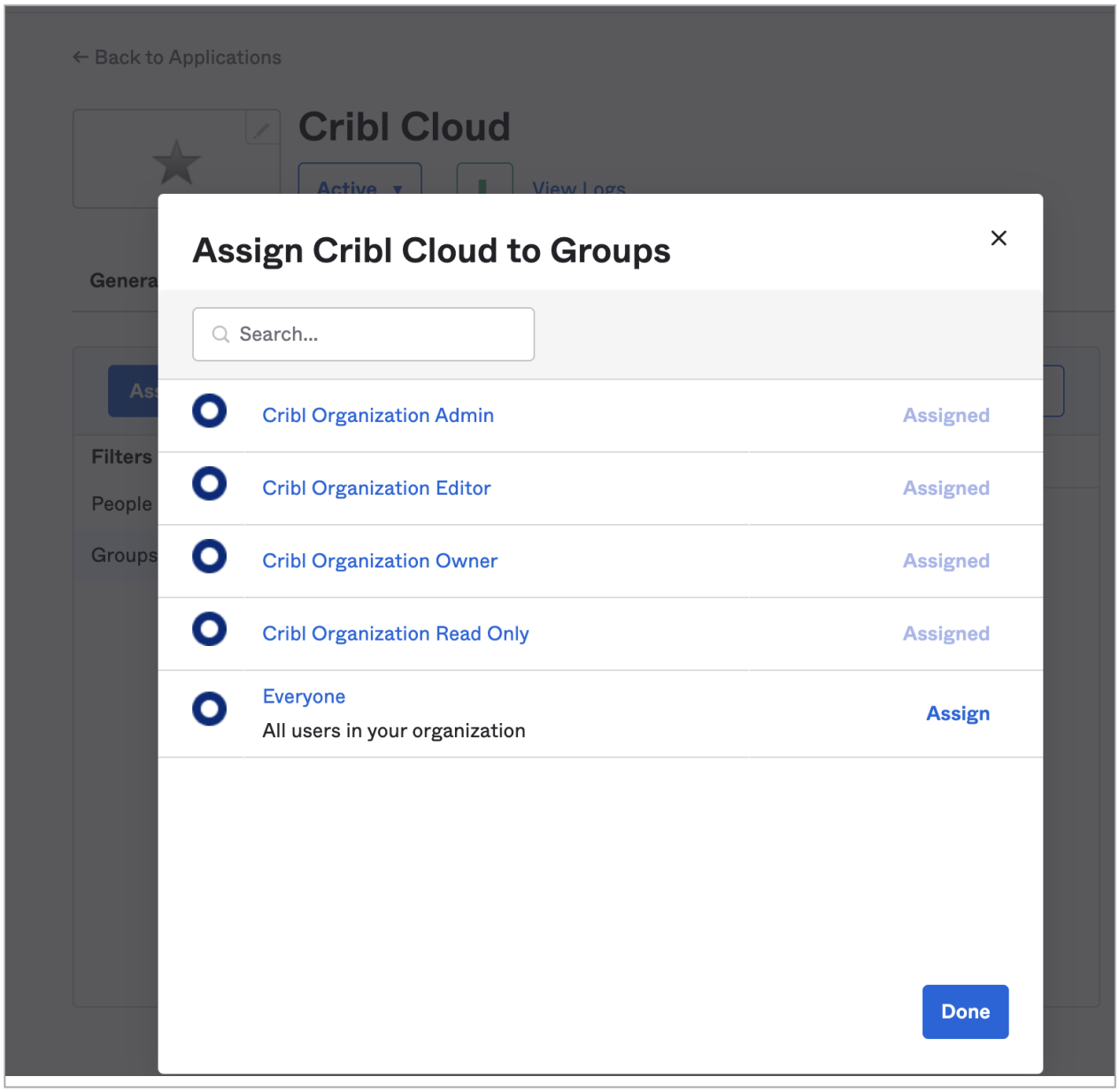
6. In the **URL** field, enter `https://portal.cribl.cloud/login?connection=<organizationID>` (where `<organizationID>` is your Cribl.Cloud Organization's ID).



Creating a bookmark app in Okta

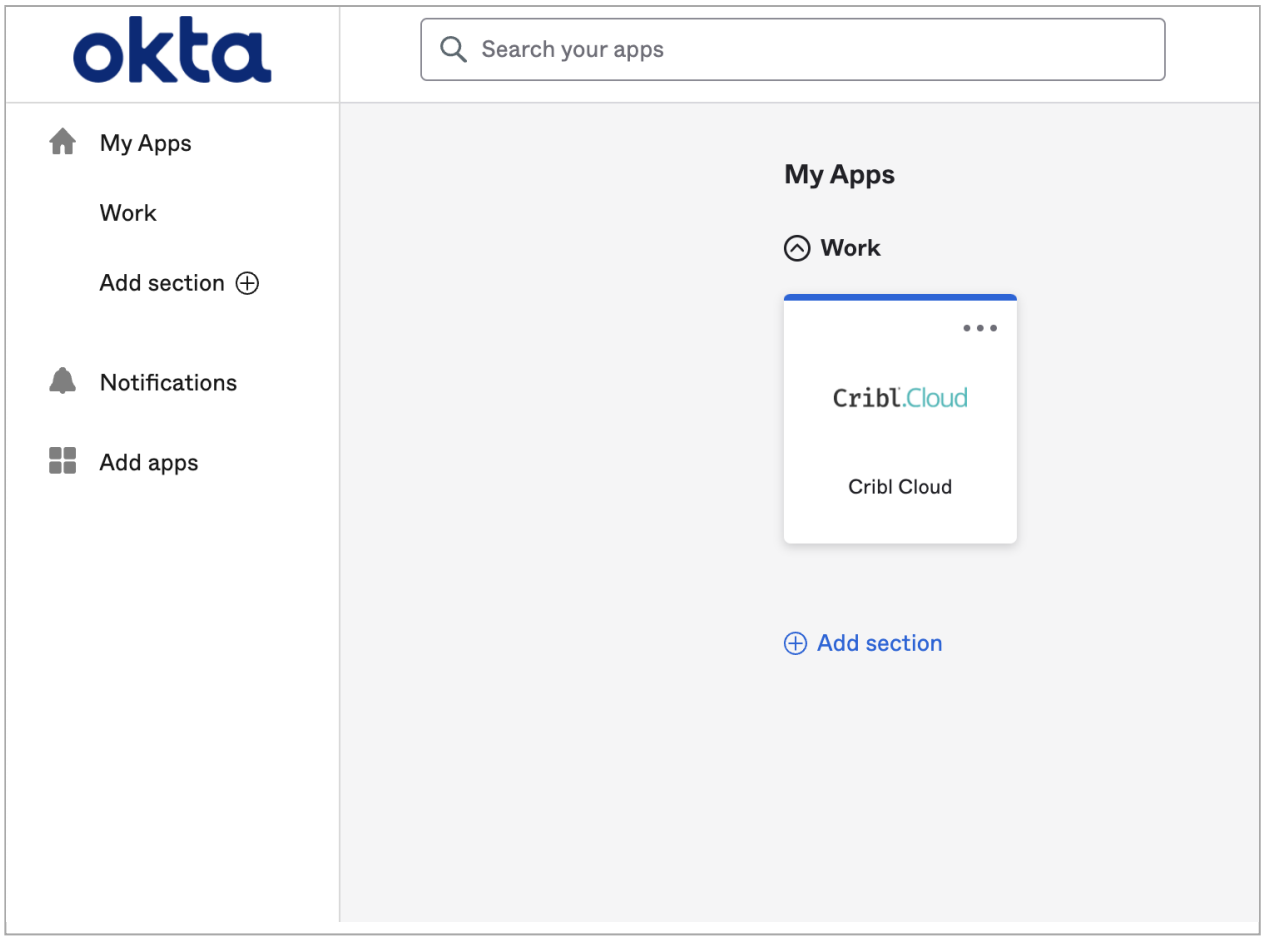
7. Click **Done**.

8. Click the **Assign** button, and assign all of the Cribl.Cloud groups to the application.



Assigning groups to the bookmark app

9. The Cribl.Cloud chiclet should now be available for all users in the Cribl groups you've assigned.



Okta Cribl.Cloud chiclet example

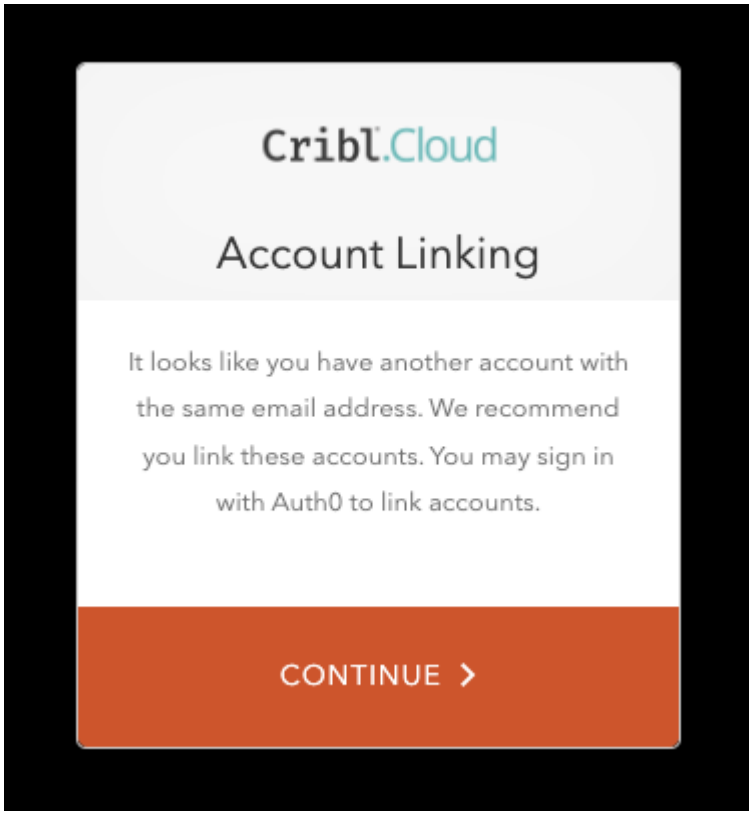
Link Existing Users

To ensure that your Cribl.Cloud Organization's local users have a smooth transition to SSO, see Final SSO Steps & Troubleshooting.

2.2.6. Final SSO Steps & Troubleshooting

Whether you're integrating with OIDC or SAML, there's one more step for users who had an existing username/password-based login on Cribl.Cloud before SSO was set up.

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. (This can be a multi-step flow.)



Prompt to link accounts

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, click 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.

3. Deploying Edge

3.1. Deployment Planning

Key points to factor in when planning an Edge deployment:

- Amount of data to be collected at the endpoint: This is defined as the amount of data planned to be ingested per unit of time for your metrics, logs, custom command outputs, etc. E.g., how many MB/s or GB/day?
- Amount of processing: This is defined as the amount of processing that will happen on incoming data. E.g., are there a lot of transformations, regex extractions, parsing functions, field obfuscations, etc.?
- Routing and/or cloning: Is most data going to a single destination, or is it being cloned and routed to multiple places? This is important, because destination-specific serialization tends to be relatively expensive.
- Deploying onto servers with no internet access: If you plan to deploy Edge Nodes on air-gapped on-prem servers (i.e., servers with no internet access), you must ensure that every Edge Node can communicate with a Leader that is on-prem (i.e., not in Cribl.Cloud).

Type of Deployment

- Use Cribl Cloud to quickly launch a Cribl-hosted deployment of the combined Cribl applications suite (Edge, Stream, and Search). With this option, Cribl assumes responsibility for provisioning and managing all infrastructure, on your behalf.

- Use Single-Instance Deployment when incoming data volume is low, and/or amount of processing is light.

- Use Fleet Management to accommodate increased load. See Add/Update Edge Nodes to streamline Edge Nodes' deployment 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.

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 Workers)
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.

Linux Requirements

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

Tested Platforms (Linux)

- **OS (Intel Processors)**:
  - Ubuntu 16.04+, Debian 9+, RHEL 7+, CentOS 7+, SUSE Linux Enterprise Server 12+, Amazon Linux 2014.03+.
- **OS (ARM64 Processors)**:
  - Ubuntu (14.04, 16.04, 18.04, and 20.04), CentOS 7.9, and Amazon Linux 2.

Browser Support

- Firefox 65+, Chrome 70+, Safari 12+, and Microsoft Edge.

Leader Scalability

Our above System Requirements suffice for Leaders handling up to 3,000 Edge Nodes. To support more than 3,000 Nodes, consider the following additional guidelines for the Leader's host:

- Deploy one CPU for every 3,000 Edge Nodes.
- Configure one Connection Process for every 3,000 Edge Nodes.

- Configure a separate Fleet hierarchy (Top-level Fleet including all SubFleets) for a maximum of 3,000 Edge Nodes. Do not push new configurations to more than one Fleet at a time.
- Increase memory by 2GB RAM for every 3,000 Edge Nodes.
- Increase memory by 2MB RAM per Edge Fleet.
- In the current release, each Leader can support a hard maximum of 15,000 Edge Nodes.

> Cribl University offers a course titled Cribl Edge Architecture: Architecture & Sizing that provides additional best practices and guidance. To follow the direct course link, first log into your Cribl University account. (To create an account, click 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.

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.

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

Edge Nodes are limited to a single process, and will consume **in the very worst case** 1 CPU. 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 (e.g., `yum install git`).
- Open the initial ports listed below:

DEFAULT PORT	PURPOSE
TCP:4200	Heartbeat/Metrics
TCP:9000	Cribl UI

- Download, install, and launch Cribl Edge (Linux, Windows).
- Configure as a Leader.
- Enable Start on Boot.
- Install License.
- Change the default auth token value.

3. Configure Edge Nodes

- Enable GUI Access. Administrators will need to connect to the following port on each Node:

DEFAULT PORT	PURPOSE
TCP:9420	Cribl UI

- 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 top nav, click **Manage** and 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 Connection Processes

To configure the connection processes for your Edge Nodes:

1. From the top nav, select **Settings** > **Global Settings** > **Service Processes** > **Services**.
2. Confirm that **Connections listener number of processes** is set to its default `1`.

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 on a Cribl Stream Leader Node, 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 root user:

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

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>
   [options] [args]

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

   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]
   [args]

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

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.

Cribl University offers a course titled How to Install & Configure Edge on Linux that illustrates the installation steps. To follow the direct course link, first log into your Cribl University account. (To create an account, click 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.

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, click 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.

```
[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
```

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_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.

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. 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.

3.2.4. System Proxy Configuration

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.

> **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. E.g., 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:

```
[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>"
```

`NO_PROXY` Usage

- Cribl recommends including the Leader Node's host name in the `NO_PROXY` list.
- Within the list, separate the host/domain names with commas or spaces.

- Optionally, each host/domain entry can be followed by a port. If specified, the port must match. If not specified, the protocol's default port is assumed.
- If specified, subdomain names must match. E.g., `NO_PROXY=foo.example.com` will send requests directly to https://foo.example.com, but https://bar.example.com requests will go through the proxy.
- You can use leading wildcards like `NO_PROXY="*.us, .org"`.
- `NO_PROXY="*"` disables all proxies.
- `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
- AWS Kinesis 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.

3.3. Installing Cribl Edge on Windows

You can install Cribl Edge on Windows Server 2016, 2019, or 2022. To start:

1. Set a compatible browser as your default browser: Microsoft Edge, Firefox, or Chrome. (Cribl Edge is not compatible with Internet Explorer.)
2. Ensure that the required ports are available (see Network Ports).
3. From Cribl's Download page, set the **Software** drop-down to either the `.exe` or `.msi` installer.
4. Click **Download Now** to get the selected installer.

You can also concatenate and copy/paste the bootstrap script in your command prompt to add Windows Node.

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.

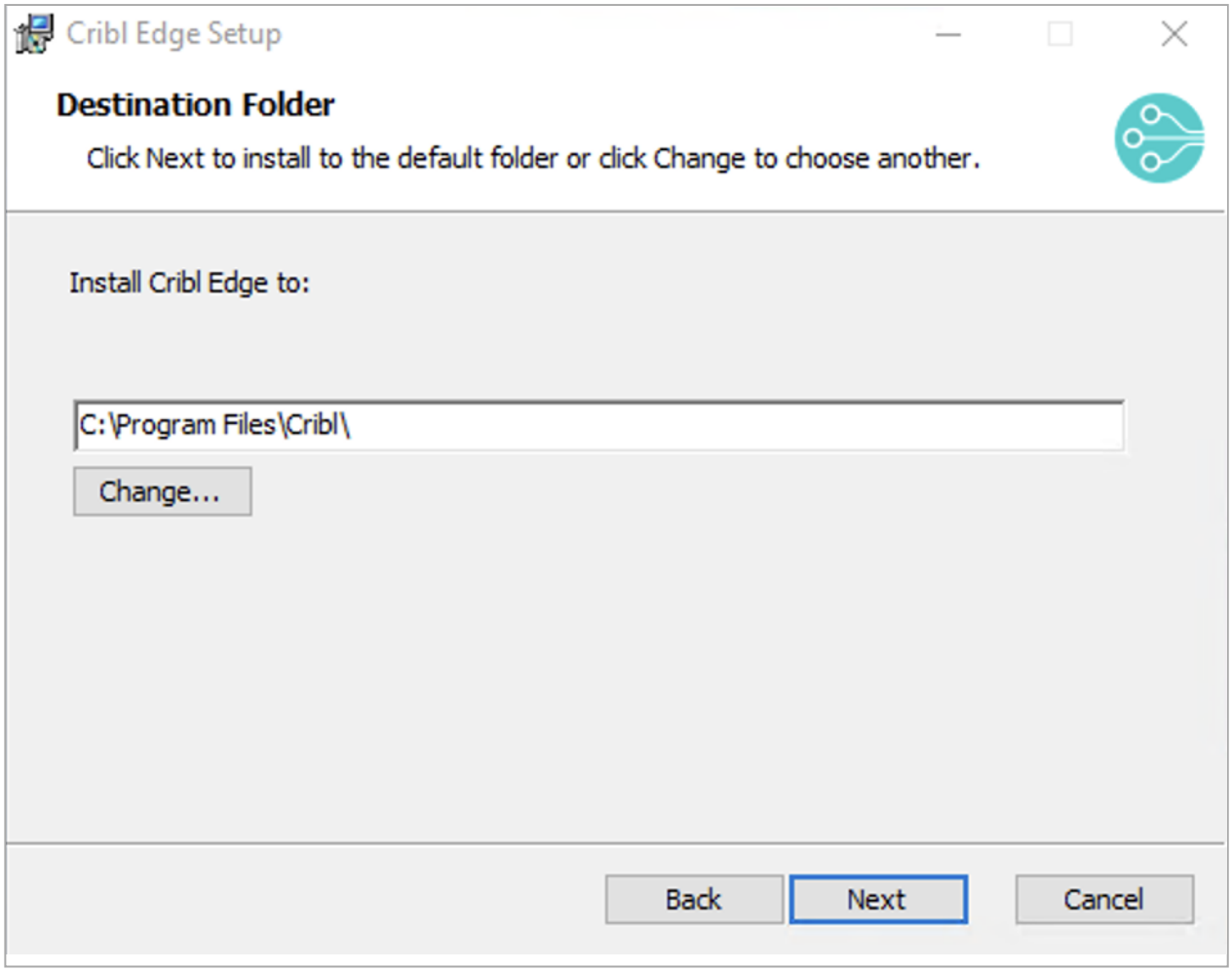
> Please see Known Issues for any current limitations on automatic version upgrades.

Using the Wizard

1. Double-click the Cribl Edge `.msi` to launch the Cribl Edge Setup Wizard. Click **Next** to start.

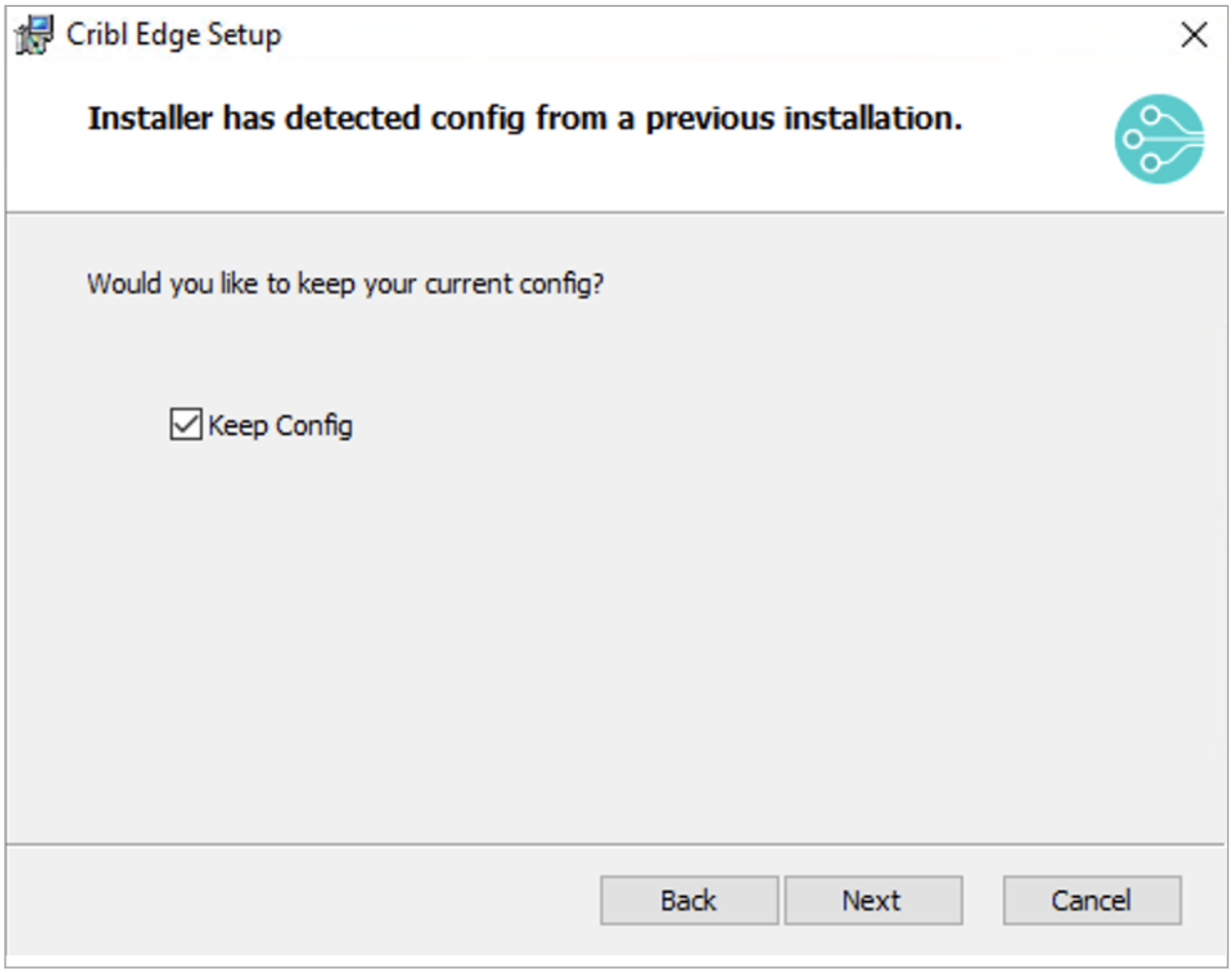
2. Read the **End-User License Agreement** and check the box to accept the terms. For the most current copy and details, see Terms of Service. Click **Next** to continue.

3. On the wizard's next page, confirm or change the **Destination Folder**. (The default path will expand as `C:\Program Files\Cribl\bin`). Click **Next** to continue.



Select location

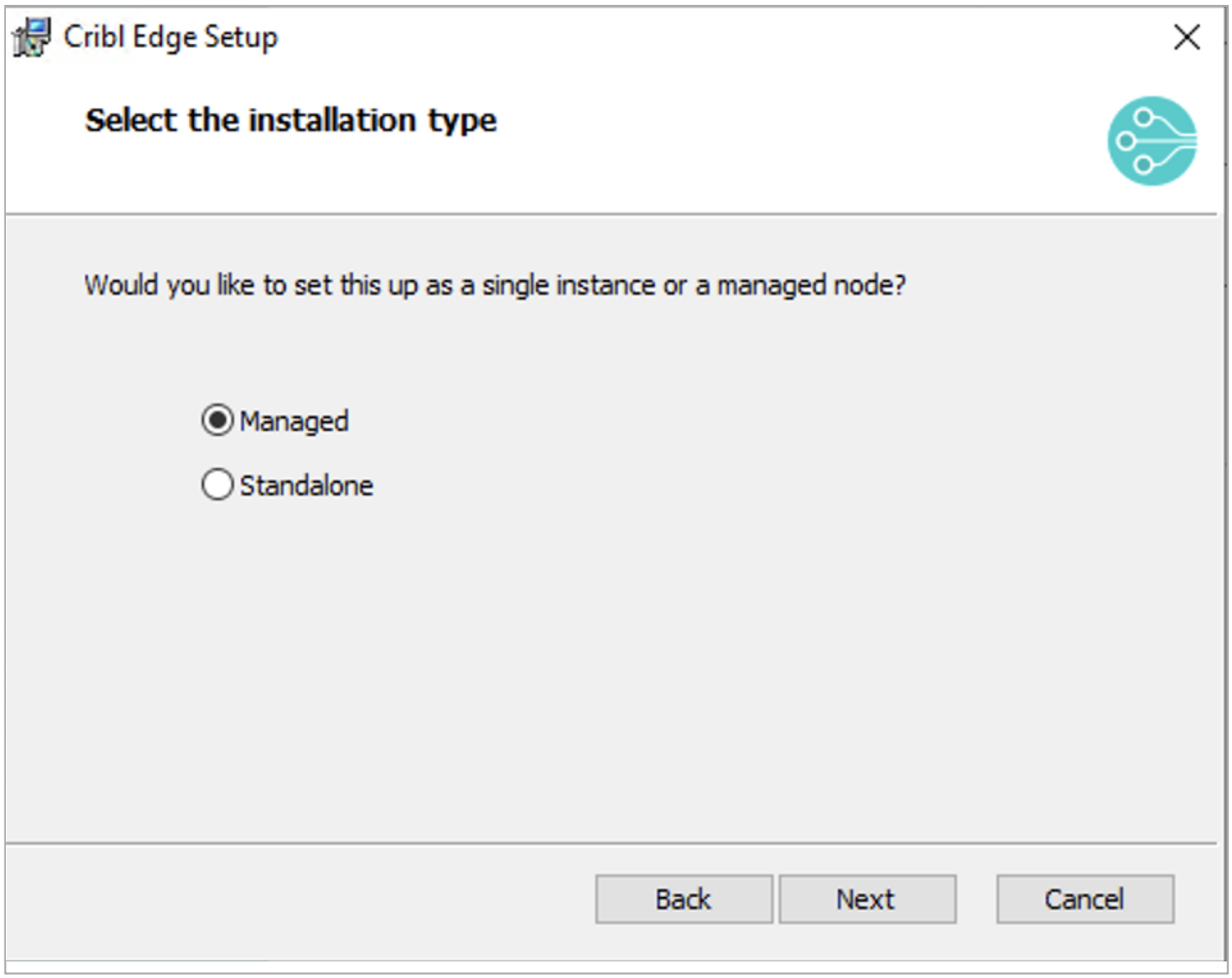
4. If the installer detects a config file from a previous installation, (i.e., prior to v.4.1), the modal will ask you if you want to keep your current config. Check the box next to `Keep Config` to persist your previous configurations stored in the `instance.yml` file. Click **Next** to continue.



Keep Config

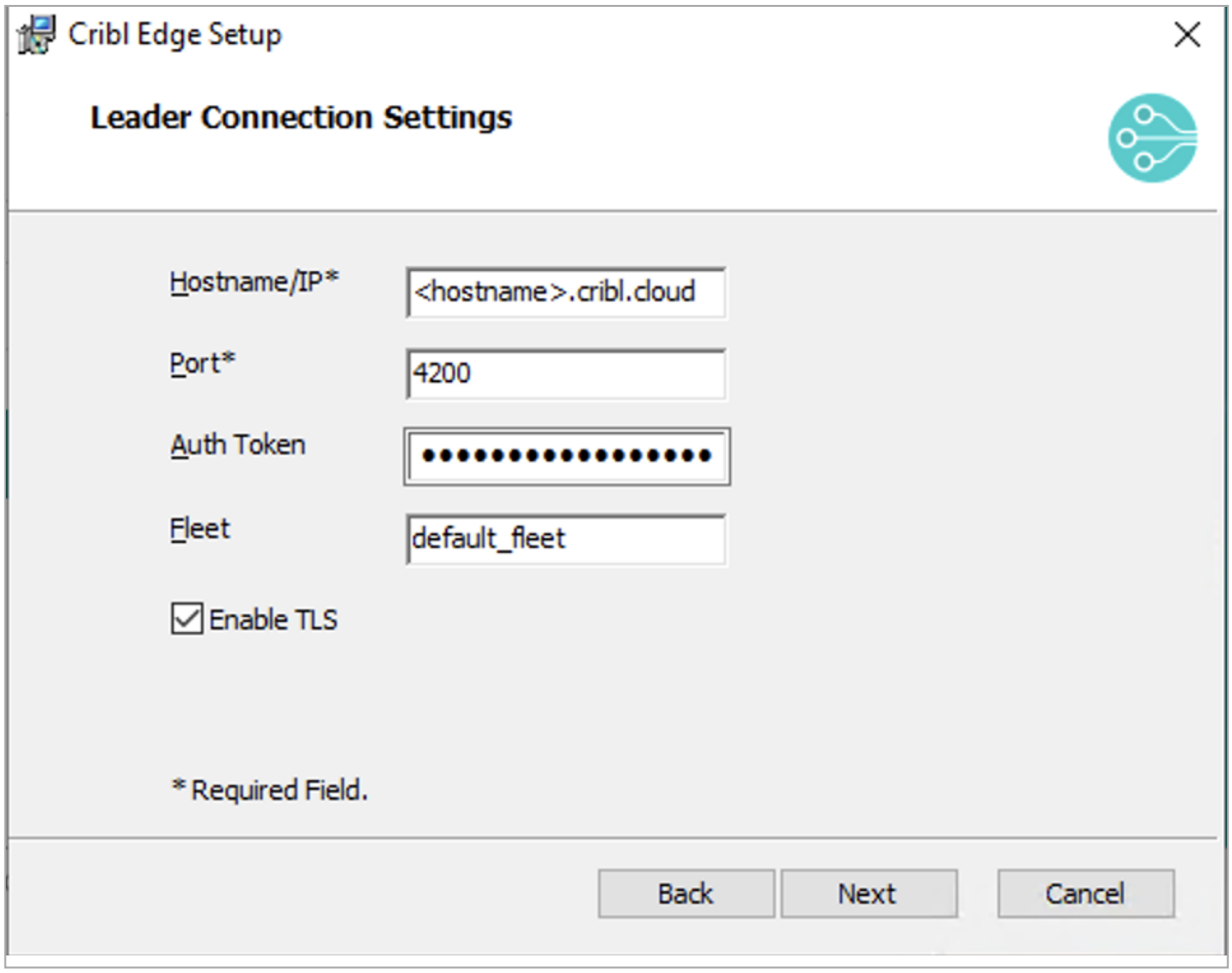
5. If you chose to keep your previous configuration, the installer will skip this option to **Select the installation type**. Select either **Managed** or **Standalone** as the installation type.

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



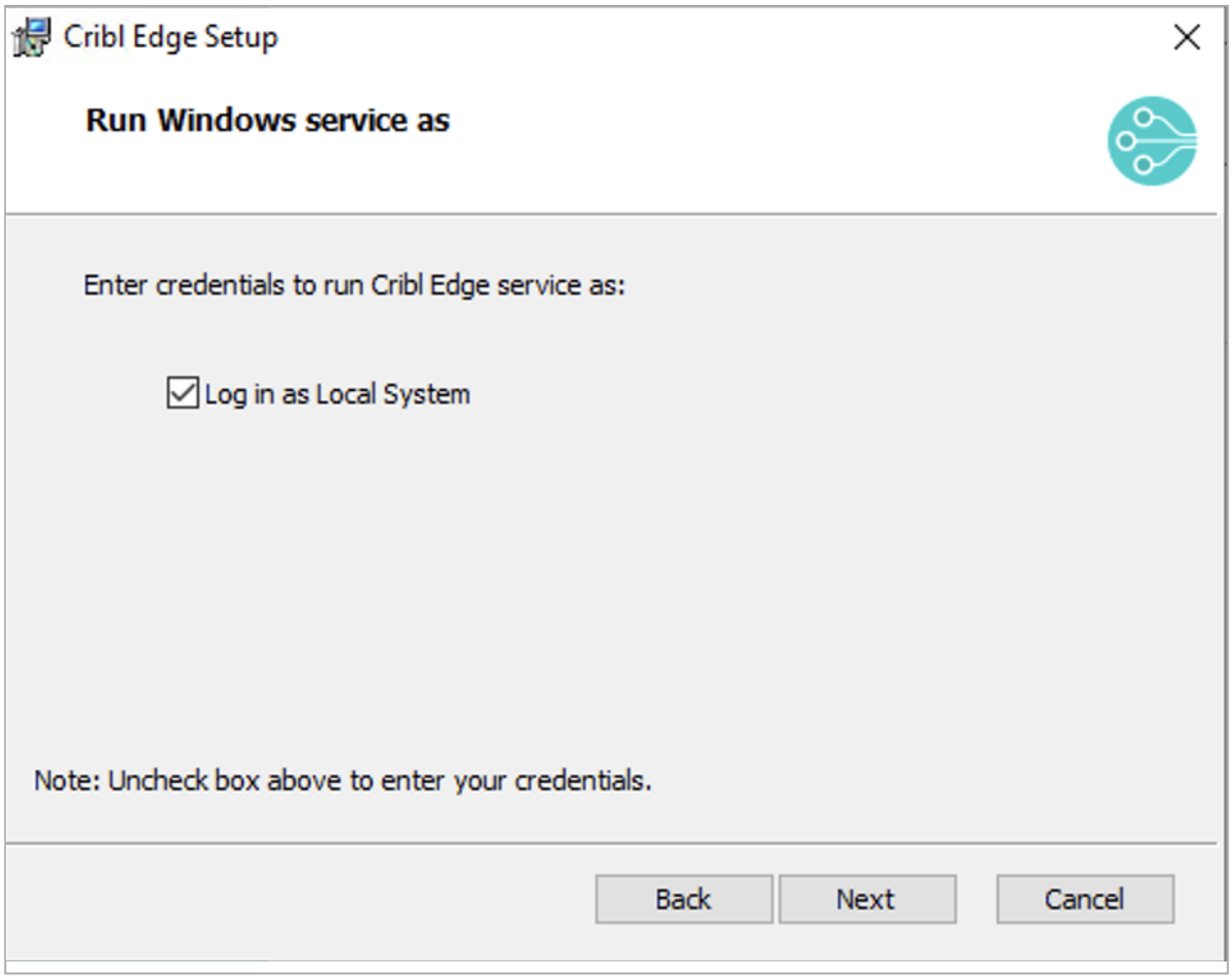
Select installation type

6. If you select **Managed** mode, enter the Leader URI (`<Leader-hostname-or-IP address>`), Auth Token, and optionally the Fleet.

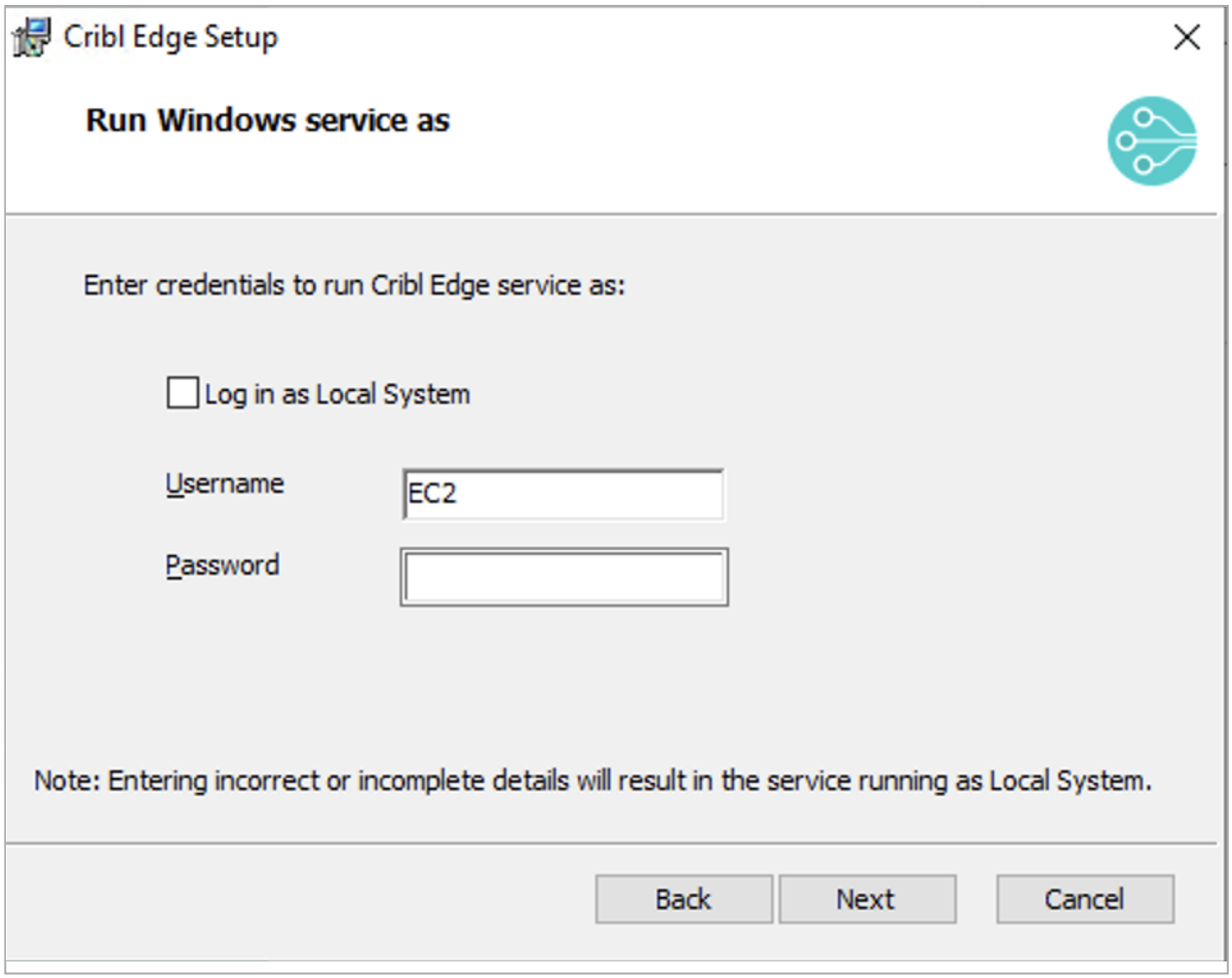


Connection Details

7. Optionally, **Log in as Local System** is checked by default. Uncheck it to 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**. Click **Next** to continue.

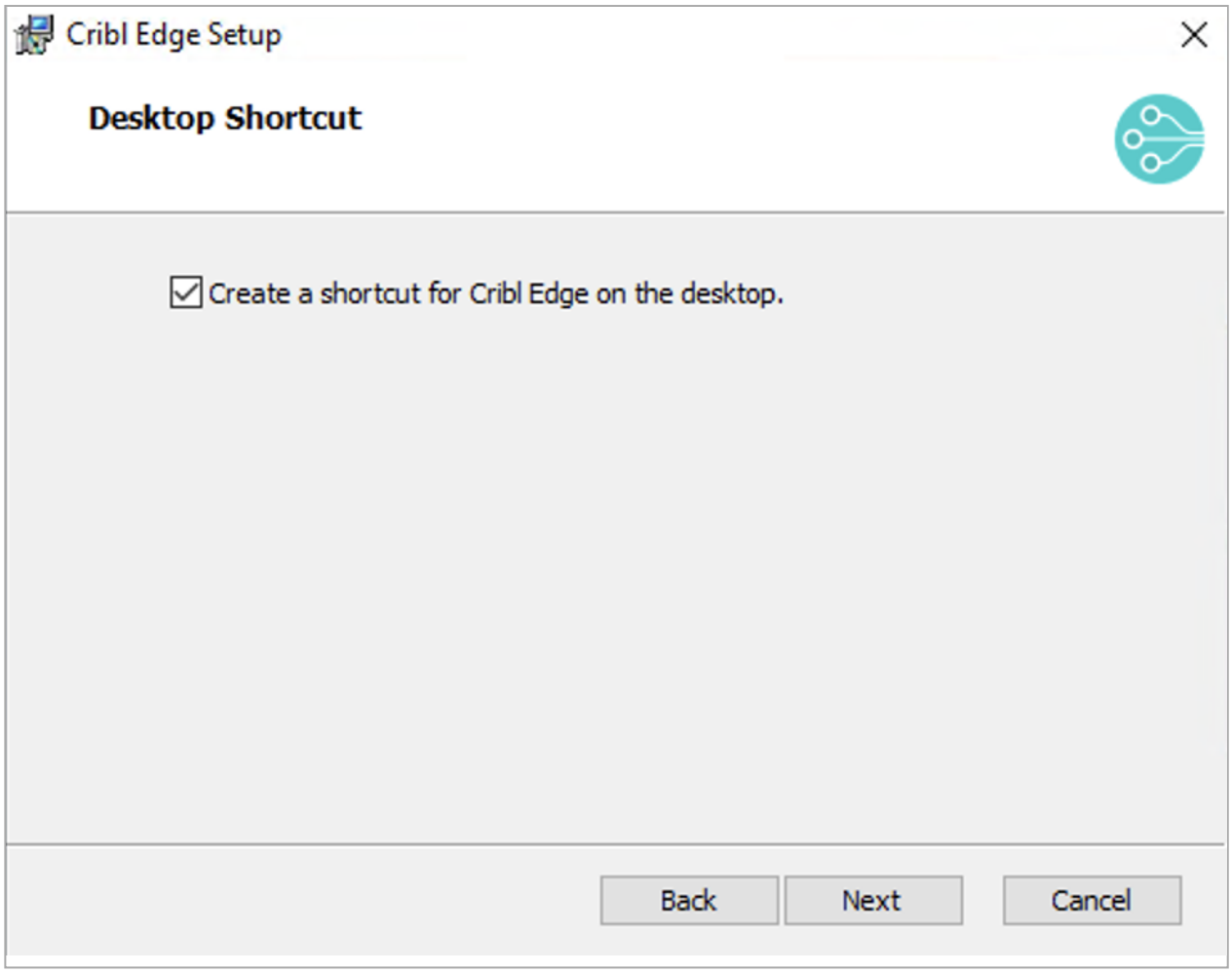


Log in as Local System



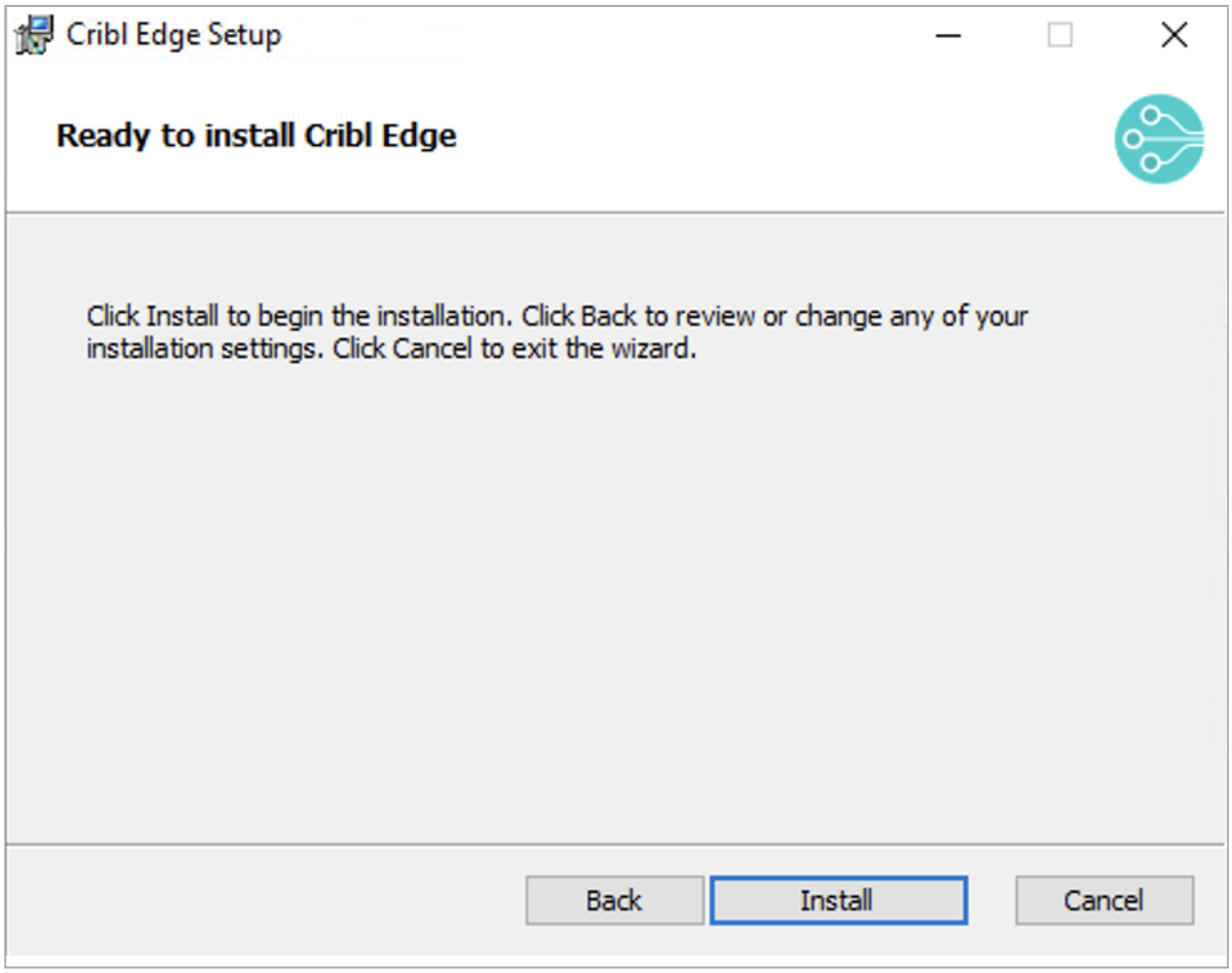
Use Credentails

8. Optionally select **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`.

Retrieving 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.

Using the MSI Installer

From the command line, you can install Cribl Edge as a single instance or as a Managed Node.

Installing 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.

Installing 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=
<yourhostname> PORT=4200 AUTH=myAuthToken FLEET=myfleet
```

For Cribl Edge v.4.1.0 or later, use the flag `COPYDATA=1` to persist data between installs.

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

For prior versions, use the flag `KEEPDATA=1` to persist data between installs.

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
PORT=4200 AUTH=myAuthToken FLEET=myfleet COPYDATA=1
```

Enabling 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 Connecting to the Leader Securely.
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
```

Troubleshooting and Logging

Here are some helpful tips:

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

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\CriblInstall.log"
```

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
PORT=4200 AUTH=myAuthToken FLEET=myfleet
""
```

> Cribl University offers a course titled How to Install & Configure Edge on Windows that illustrates the installation steps. To follow the direct course link, first log into your Cribl University account. (To create an account, click 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.

3.4. 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.

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

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:

Dockerfile

```
FROM cribl/cribl:latest
RUN apt-get update && \
    apt-get -y upgrade dpkg
```

> Cribl Edge v.4.1.x and later supports both Docker and `containerd` runtimes.

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 encounter spurious health-check failures, change that **Host** setting to `0.0.0.0` (rather than the default `127.0.0.1`, which can block health checks).

> Cribl University offers a course titled How to Install & Configure Edge on Docker that illustrates the installation steps. To follow the direct course link, first log into your Cribl University account. (To create an account, click 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 v.4.1.x and later 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
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
```

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

By default, Cribl Edge's 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.

By default, the API Server is configured to listen on the loopback interface (`127.0.0.1`). Change the listening interface address to `0.0.0.0` to 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.

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
```

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_FOOTGUN` - set to `true` to enable resource-intensive, potentially risky modes of the Kubernetes Metrics and Kubernetes Logs Sources.
- `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.

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.

Cribl University offers two courses that provide a good overview of working with Kubernetes: How to Install & Configure Edge on Kubernetes and Collecting Data from Kubernetes. To follow the direct course link, first log into your Cribl University account. (To create an account, click 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.

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.

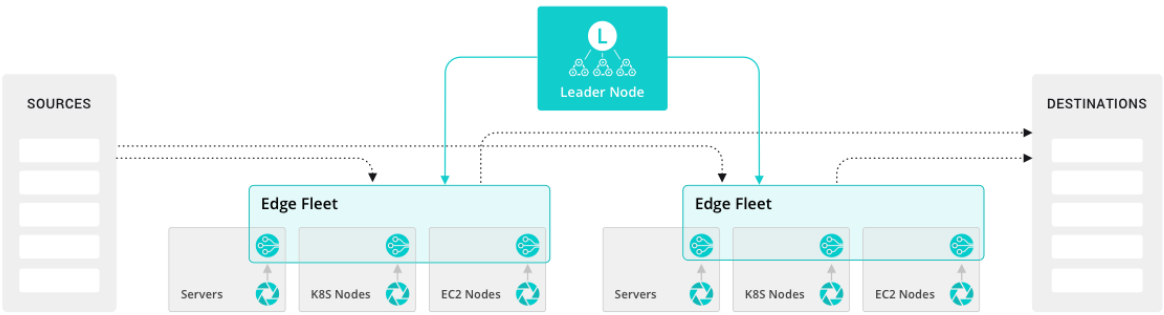
Worker Process – a Linux process within a Single Instance, or within Edge Nodes, that handles data inputs, processing, and output. The process count for Edge Nodes is constrained to 1.

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**: Firefox 65+, Chrome 70+, Safari 12+, 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

Fleets & their corresponding Subfleets allow you to author and manage configuration settings for a particular set of Edge Nodes. The Cribl Edge UI offers you the flexibility to group edge nodes into logical Fleets and Subfleets so they can share and reuse configurations.

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

To create a new Fleet, first click the top nav's **Manage** tab. Then, from the resulting **Fleets** lower tab, click **New Fleet**.

To add a new Fleet, enter a descriptive **Fleet name**, and optionally, a relevant **Description**. To enable teleporting to the Edge Nodes via the Leader's UI, set **Enable teleporting to Nodes** to **Yes**, then click **Save**.

Subfleets and Inheritance Configurations

You can organize your Fleets and Subfleets into a hierarchy of configuration layers from the top level down. 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.

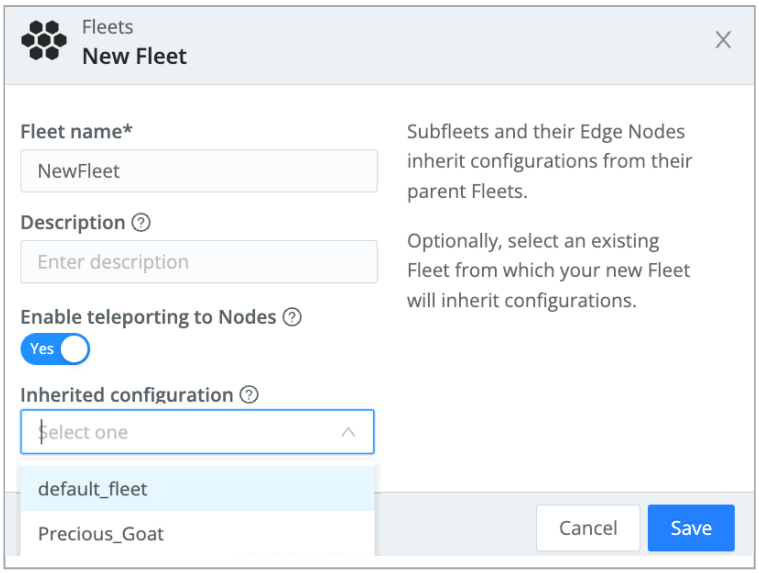
There are several ways you can build the hierarchy:

- Organizational.
- Geographical (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.

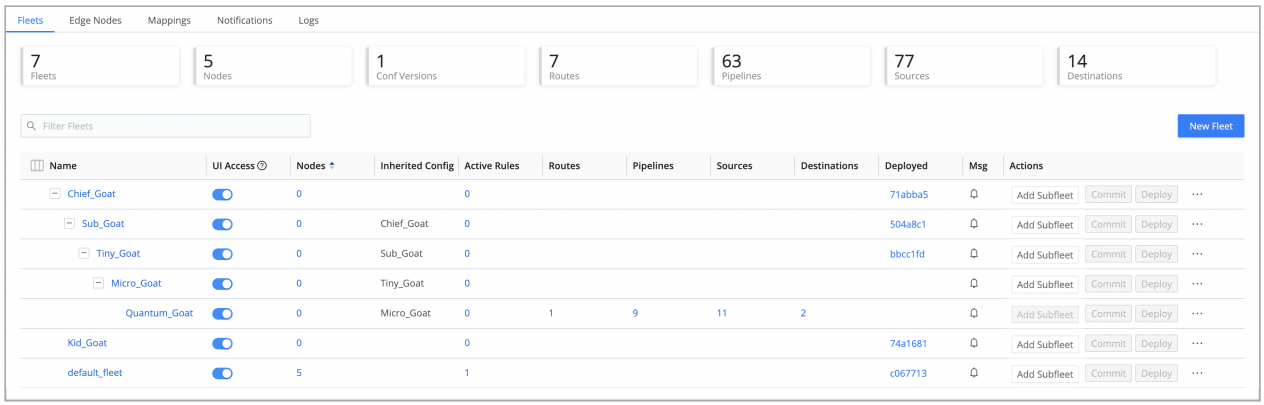
To add a Subfleet, enter a descriptive **ID**. Next, use the **Inherited configuration** drop-down to select the existing Fleet from which it will inherit its configuration. Then click **Save**.

You can also add a new Subfleet from a parent Fleet, by clicking its **Actions** column's **Add Subfleet** button.



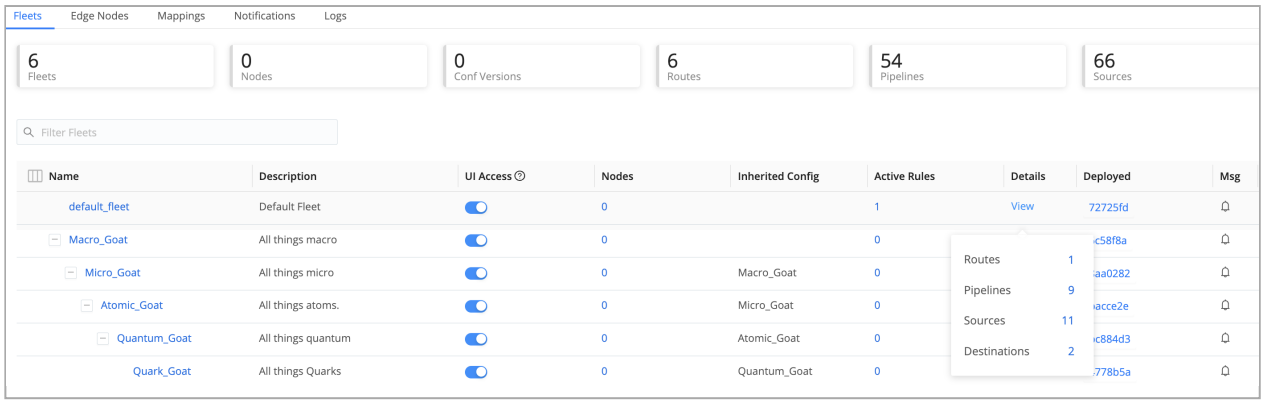
Add a new Fleet

You can similarly create a child Fleet in a selected Subfleet. The maximum depth for inheriting configurations is 5 levels, at which point the **Add Subfleet** button will be disabled.



Subfleet's maximum depth

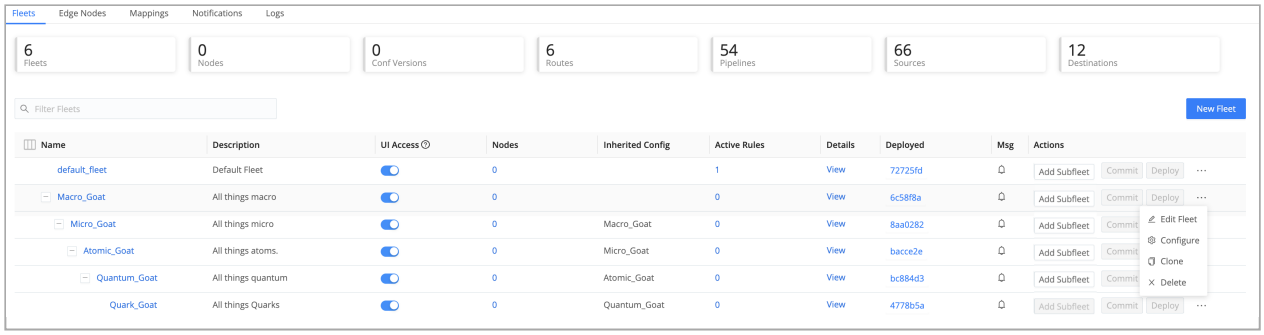
In a Fleet's or Subfleet's **Details** column, click **View** to display the Fleet's number of Routes, Pipelines, Sources, and Destinations.



Configure, Clone, and Delete Fleets and Subfleets

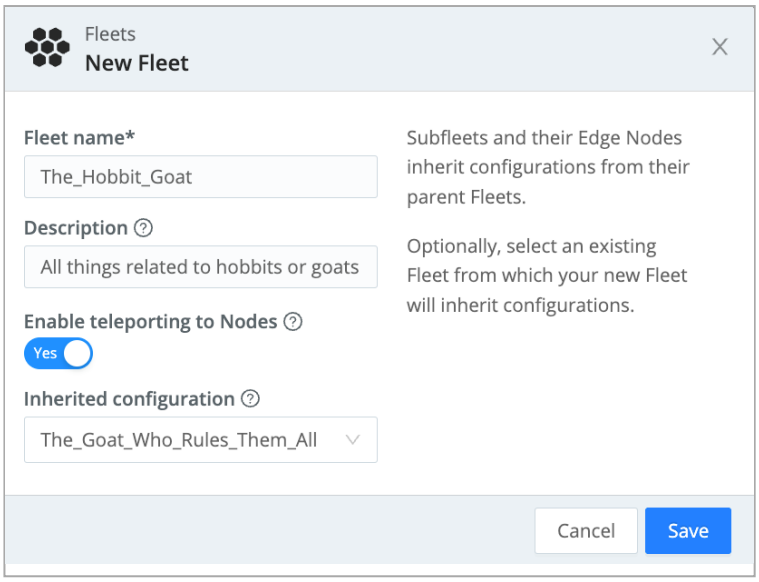
> On Cribl.Cloud, you'll also see a **Group Type** column, showing whether the deployment is **Hybrid** or **Cloud**.

Click the Fleet's or Subfleet's ••• (Options) menu to display the **Edit Fleet**, **Configure**, **Clone**, and **Delete** options.



Edit, Configure, Clone, and Delete Fleets and Subfleets

Cloning a Fleet or Subfleet displays the **Clone** modal. Cloning a Fleet copies the Fleet's settings, while cloning a Subfleet inherits the parent's configurations.



Clone a Subfleet

To delete a Fleet or Subfleet, click the ••• (Options) menu's **Delete** option. 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 requires an Edge Enterprise or Standard license.

Inheritance Limitations

The inheritance configurations are currently subject to the following limitations. We will update this section as we enable more features:

- Inherited Packs are not visible in the Packs list. You can access inherited Packs by selecting **More** > **Data Routes**, then opening a Route's **Pipeline** drop-down.
- Inherited Lookups and Samples are not visible in their respective lists.

However, when you deploy the configuration changes to the Edge Nodes, you can teleport into a Node and view the inherited Packs, Lookups, and Samples in their respective lists.

> Cribl University offers two courses that provide a good overview of Fleets: Fleet Administration & Management and Fleet Hierarchy. To follow the direct course link, first log into your Cribl University account. (To create an account, click 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.

Configuring a Fleet

Navigate to a Fleet/Subfleet, click the ••• (Options) menu to display the **Configure** option. Click **Configure** to display an interface for authoring and validating its configuration. You can configure everything for this Fleet as if it were a single Edge instance – using a similar visual interface.

Can't Log into the Edge Node as Admin User?

To explicitly set passwords for Fleets, see User Authentication.

Committing and Deploying Fleet Configurations

To make new configurations visible and available to Subfleets, you must first **Commit** the changes within parent Fleets' configurations.

On Cribl Edge v.4.1.x and later, when you commit changes to a parent Fleet, you now have the option to **Commit and Deploy with Subfleets**. When you **Deploy with Subfleets**, the Subfleets will receive the new inherited configuration. For guidelines on deploying to a large number of Edge Nodes, see Leader Scalability.

Deploy with Subfleets

Recovering a Fleet

If you delete a Fleet, you lose the context required to access the Version Control menu. As a result, you also lose the option to recover the Fleet before it was deleted.

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

- Creating a New Fleet with the Same Name
- Temporarily Stopping the Cribl Server

Creating a New Fleet with the Same Name

1. Create a new Fleet with the same name as the one that was deleted. This action will restore the deleted Fleet to the system.

2. Select a previous version of the deleted Fleet from the Version Control menu if you previously committed the deletion of your Fleet.

If you did not previously commit the deletion of your Fleet, the system will automatically restore it.

Temporarily Stopping the Cribl Server

1. Stop the Cribl server.

2. Identify the last commit before the Fleet was deleted.

3. Revert to that commit.

4. Restart the Cribl server.

Mapping Edge Nodes to Fleets

Mapping Rulesets are used to map Edge Nodes to Fleets. Within a ruleset, a list of rules evaluate Filter expressions on the information that Edge Nodes send to the Leader.

Only one Mapping Ruleset can be active at any one time, although a ruleset can contain multiple rules. At least one Fleet should be defined and present in the system.

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 Edge Nodes on Multiple Platforms

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.

For mapping details, see the Mapping sections below. For what's supported on Windows, see Cribl Edge on Windows.

Creating a Mapping Ruleset

To create a Mapping Ruleset, click **Manage** on the top nav, then click the **Mappings** tab. On the resulting page, click **Add Ruleset**. Give the resulting **New Ruleset** a unique **ID** and click **Save**. Click your new ruleset's **Configure** button, and start adding rules by clicking on **Add Rule**. While you build and refine rules, the Preview in the right pane will show which currently reporting and tracked workers map to which Fleets.

A ruleset must be activated before it can be used by the Leader. 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, Edge Nodes can be configured to send a preferred Fleet name with their payload (found in payload's `cribl.group` key) See below how this ranks in mapping priority.

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

Add a Mapping Rule – Example

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

When an Edge instance runs as Leader, the following are created automatically:

- 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.

Deploying Configurations

Your typical workflow for deploying Edge configurations is the following:

1. Work on configs.
2. Save your changes.
3. Commit (and optionally push).
4. Deploy.

Deployment is the last step after configuration changes have been saved and committed. **Deploying** here means propagating updated configs to Edge Nodes. You deploy new configurations at the Fleet level: Locate your desired Fleet and click on **Deploy**.

On Cribl Edge v.4.1.x and later, when you commit changes to a parent Fleet, you now have the option to **Deploy with Subfleets**. When you **Deploy with Subfleets**, the Subfleets will receive the new inherited configuration.

Deploy with 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 into 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 under: `$CRIBL_HOME/groups/<FleetName>/local/edge`.

On the managed Edge Node, after configs have been pulled, they are extracted 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.

Environment Variables

- `CRIBL_DIST_MASTER_URL` – URL of the Leader Node. Format: `<tls|tcp>://<authToken>@host:port?group=defaultGroup&tag=tag1&tag=tag2&tls.<tls-settings below>`.

  Example: `CRIBL_DIST_MASTER_URL=tls://<authToken>@leader:4200`

Parameters:

  - `group` – The preferred Worker Group 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 to a Worker Group.
  - `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_NOAUTH` – Disables authentication. Careful here!!

- `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. Format: `<socks4|socks5>://<username>:<password>@<host>:<port>`. Only `<host>:<port>` are required.

  The default protocol is `socks5://`, but you can specify `socks4://proxyhost:port` if needed.

  To authenticate on a SOCKS4 proxy with username and password, use this format: `username:password@proxyhost:port`. The `proxyhost` can be a `hostname`, `ip4`, or `ip6`.

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/bin/`, e.g.:

```
# cat CRIBL_HOME/bin/676f6174733432.dat

{"it":1570724418,"phf":0,"guid":"48f7b21a-0c03-45e0-a699-01e0b7a1e061"}
```

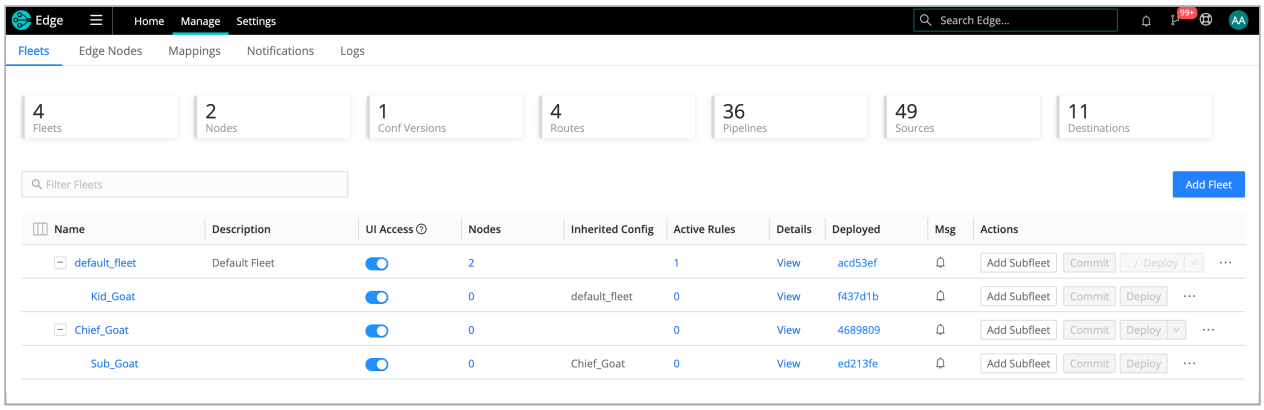
When deploying Cribl Edge as part of a host image or VM, be sure to remove this file, so that you don't end up with duplicate GUIDs. The file will be regenerated on next run.

4.2. Managing Edge Nodes

If you have an Enterprise or Standard license, you can click the **Manage** tab in the left nav to open the **Manage Fleets** page. This page has three upper tabs: **Fleets**, **Edge Nodes** and **Mappings**.

Fleets Tab

The **Manage Fleets** page provides a list of all configured Fleets and Subfleets in the instance.

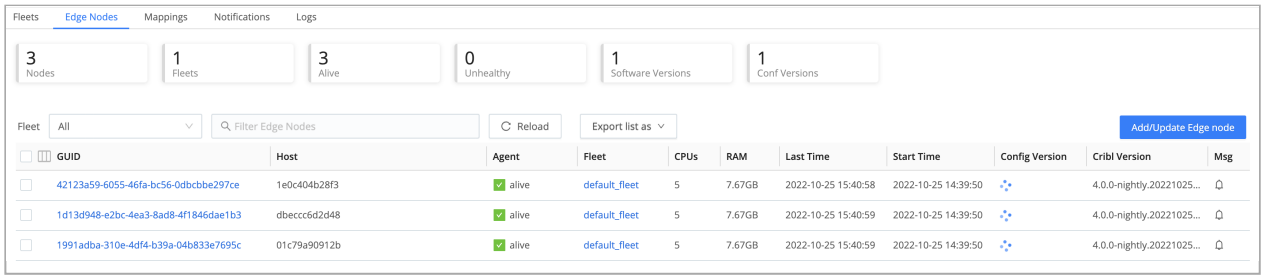


Manage Fleets

The top header now shows you the number of configured Fleets (2), along with other statistics. Clicking on the Fleet's **Name** redirects you to its Fleet Landing Page where you can explore more information for each of your configured Edge Nodes.

Edge Nodes Tab

The **Edge Nodes** tab on the **Manage Edge Nodes** page provides status information for each Edge Node in the selected Fleet.



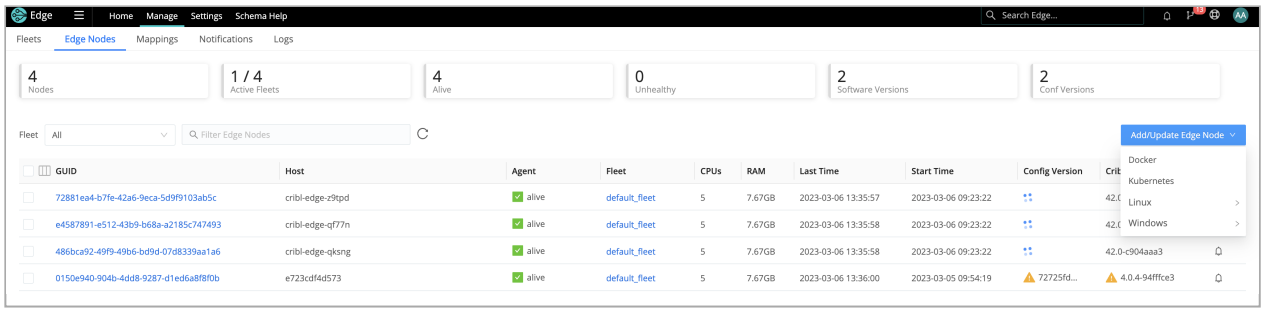
Edge Node tab

- 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 also be filtered. You can export the entire list by removing all filters.
- To display additional details and controls, click each row.
- To teleport from the Leader into the Edge Node, click the **Edge Node GUID** link.

Add/Update Edge Nodes

You can use the **Add/Update Edge Node** control at upper right to update an existing Node, or to add a new Node by generating a bootstrap script. Cribl Edge admins can use the UI to concatenate and copy/paste the bootstrap script, automating several steps below. You can use an adjacent option to grab a script that updates an Edge Node's Fleet assignment. You also have the option of generating a bootstrap script for the following Edge Nodes:

- 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.



Options to add or update an Edge Node

> 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.

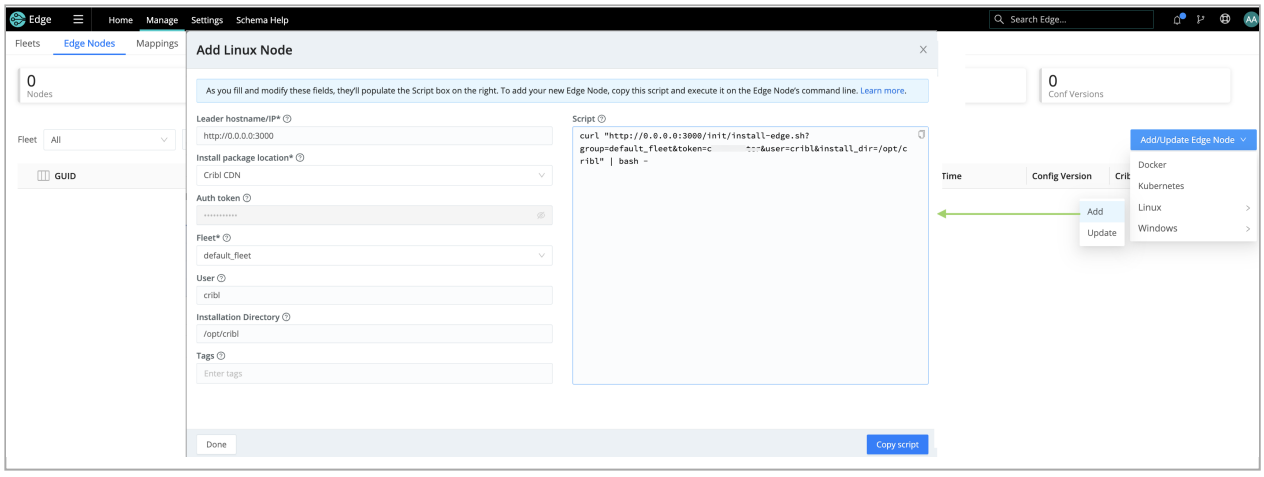
Add/Bootstrap a New Edge Node

> The details below differ slightly depending on which deployment option you select.

1. From Cribl Edges's top nav, select Manage > Edge Nodes.

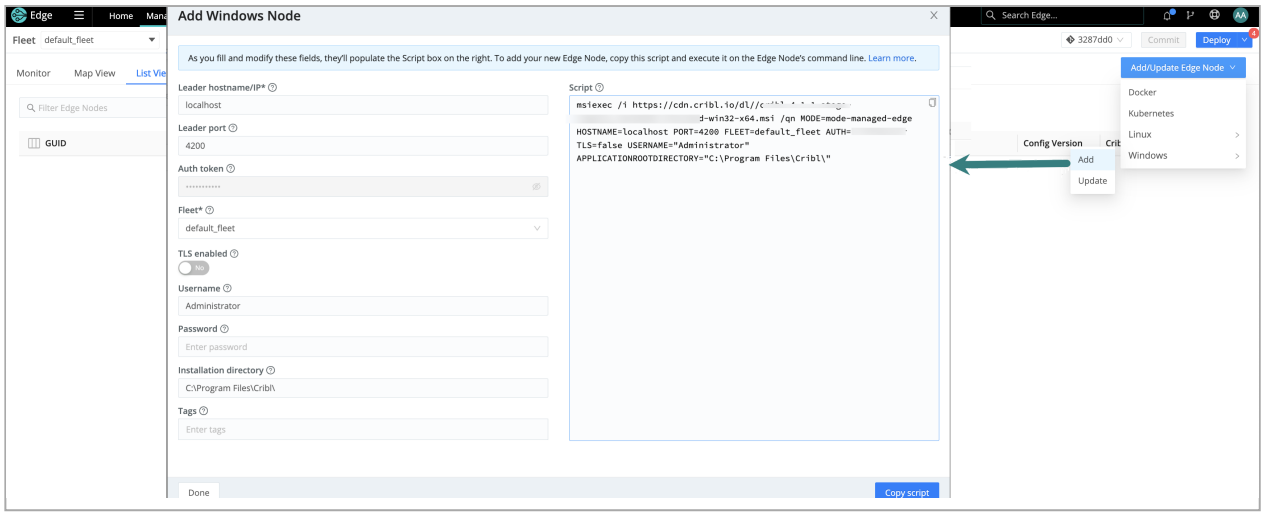
2. On the resulting Edge Nodes tab, click **Add/Update Edge Node** at the upper right.
3. Select the deployment option as shown in the composite screenshots below.
4. In the resulting modal, the **Install package location** defaults to `Cribl CDN`. If desired, change this to `Download URL`.
5. As needed, correct the target **Fleet**, as well as the **Leader hostname/IP** (URI).
6. As needed, correct the **User** to run Cribl as. Defaults to `cribl`.
7. As needed, correct the **Installation directory**. Defaults to `/opt/cribl`.
8. Optionally, add **Tags** that you can use for filtering and grouping in Cribl Edge. Use a tab or hard return between (arbitrary) tag names.
9. Copy the resulting script to your clipboard.
10. Click either **Done** or **X** to close the modal.

Paste the script onto your Edge Node's command line and execute it, to add the Edge Node.



Generating a script to bootstrap an Edge Node on Linux

Below is a composite screenshot of adding an Edge Node on Windows.



Generating a script to bootstrap an Edge Node on Windows

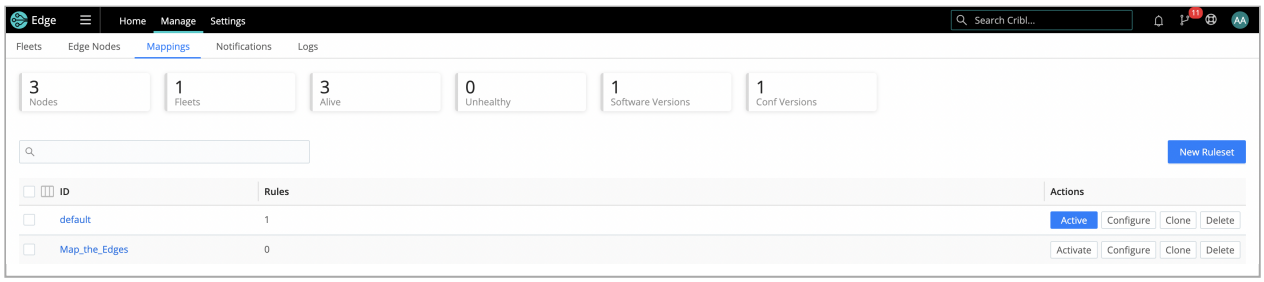
Troubleshooting the Display

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

- Edge Nodes that miss 5 heartbeats, or whose connections have closed for more than 30 seconds, will be removed from the list.
- 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.

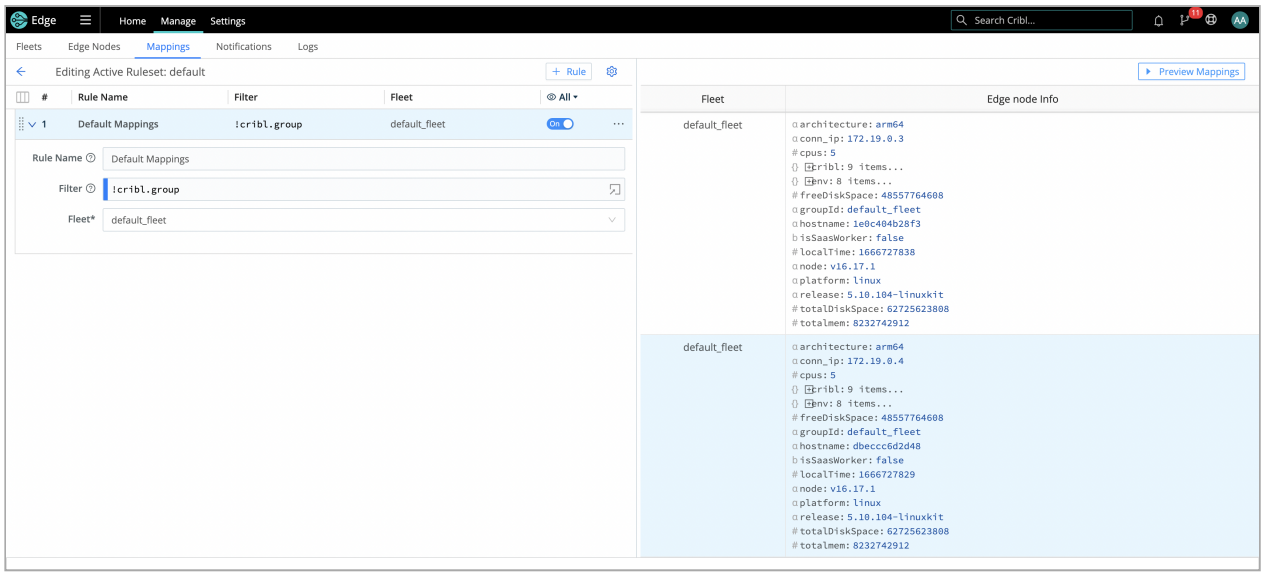
Mappings Tab

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



Mappings status/controls

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

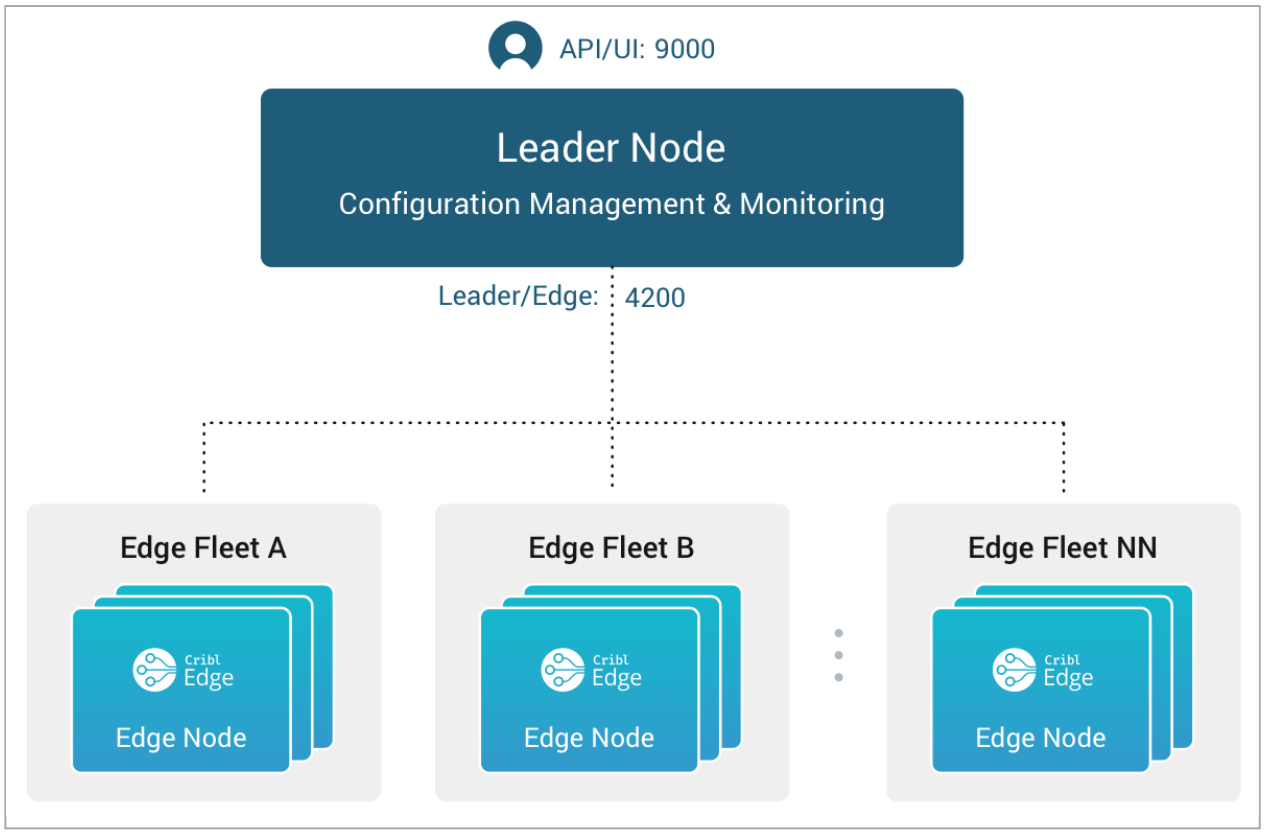


Managing Ruleset page

How Do Edge Nodes and Leader 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 (Heartbeat/Metrics/other).

Leader/Edge Node Communication

Edge Nodes will periodically (every 10 seconds) send a heartbeat to the Leader. This heartbeat includes information about themselves, 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.

Config Bundle Management

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.

4.3. Setting Up Leader and Edge Nodes

This page covers:

1. Configuring a Leader Node
2. Configuring an Edge Node

Configuring a Leader Node

You can configure a Leader Node either through the UI or through the `instance.yml` config file.

Using the UI

In **Settings** (top nav) > **Global Settings** > **Distributed Settings** > **Distributed Management** > **General Settings**, select **Mode**: **Leader**.

Next, on the **Leader Settings** left tab, confirm or enter the required Leader settings (**Address** and **Port**). Customize the optional settings if desired. Then click **Save** to restart.

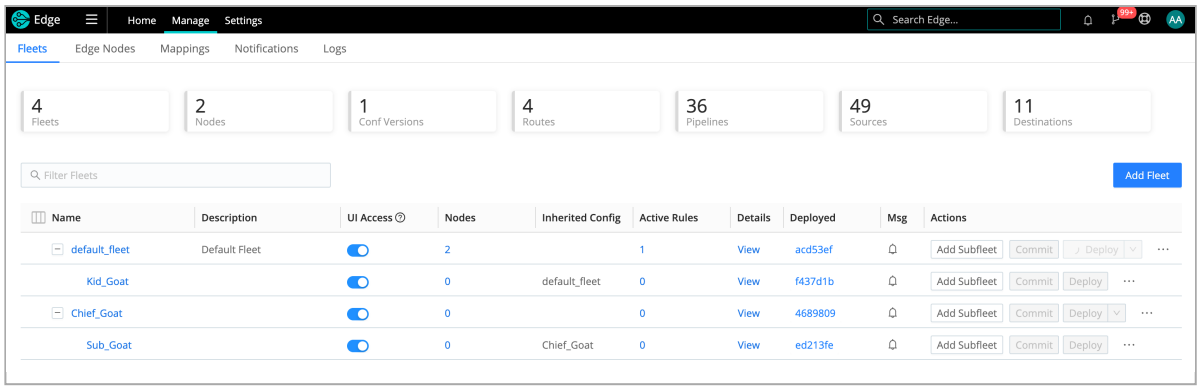
Edge Node UI Access

This useful option enables you to click through from the Leader's **Manage Edge Nodes** page to an authenticated view of each Node's UI. 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. From Cribl Edge's top nav, select **Manage**.

2. On the **Manage Fleets** page: For each desired Fleet, toggle **UI Access** to `On`.

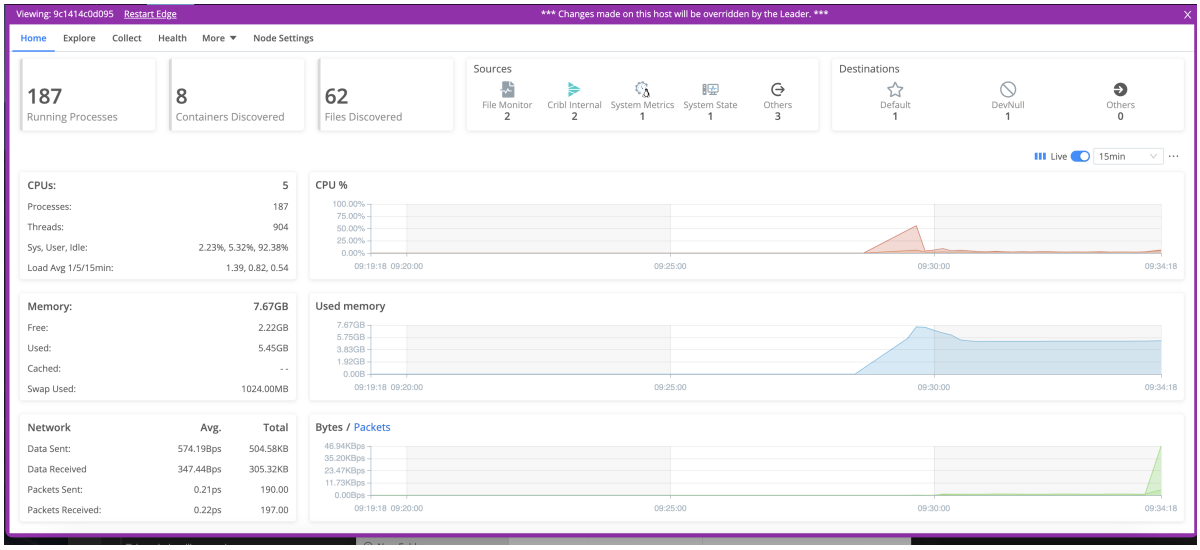


Manage Fleets

3. From the top nav, select **Edge Nodes**.

4. On the **Manage Edge Nodes** page, click the link for any Node you want to inspect.

   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.

Using YAML Config File

In `$CRIBL_HOME/local/_system/instance.yml`, under the `distributed` section, set `mode` to `master`:

```
$CRIBL_HOME/local/_system/instance.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 (e.g., `/tmp`).

Many Linux distros maintain a system cleaner service (e.g., 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 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**.

Configuring 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.

Using the UI

In **Settings** (top nav) > **Global Settings** > **Distributed Settings** > **Distributed Management** > **General Settings**, select **Mode**: `Managed Edge`.

Next, on the **Leader Settings** left tab, confirm or enter the required **Address** (e.g., `criblleader.mycompany.com`).

Customize the optional settings if desired. Then click **Save** to restart.

Using YAML Config File

In `$CRIBL_HOME/local/_system/instance.yml`, under the `distributed` section, set `mode` to `managed-edge`:

$CRIBL_HOME/local/_system/instance.yml

```
distributed:
  mode: managed-edge
  envRegex: /^CRIBL_/
  master:
    host: <master address>
    port: 4200
    authToken: <token here>
    compression: none
    tls:
      disabled: true
    connectionTimeout: 5000
    writeTimeout: 10000
  tags:
       - tag1
       - tag2
       - tag42
  group: teamsters
```

Using Environment Variables

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

```
CRIBL_DIST_MASTER_URL=tcp://criblmaster@masterHostname:4203 ./cribl start
```

For additional details, see Environment Variables.

Using 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 `-H` and `-p` parameters are required. 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 this command is issued.

4.4. Leader High Availability/Failover

To handle unexpected outages in your on-premises distributed deployment, Cribl Edge 3.5 and above supports configuring a second Leader for failover. This way, if the primary Leader goes down, Collectors and Collector-based Sources can continue ingesting data without interruption.

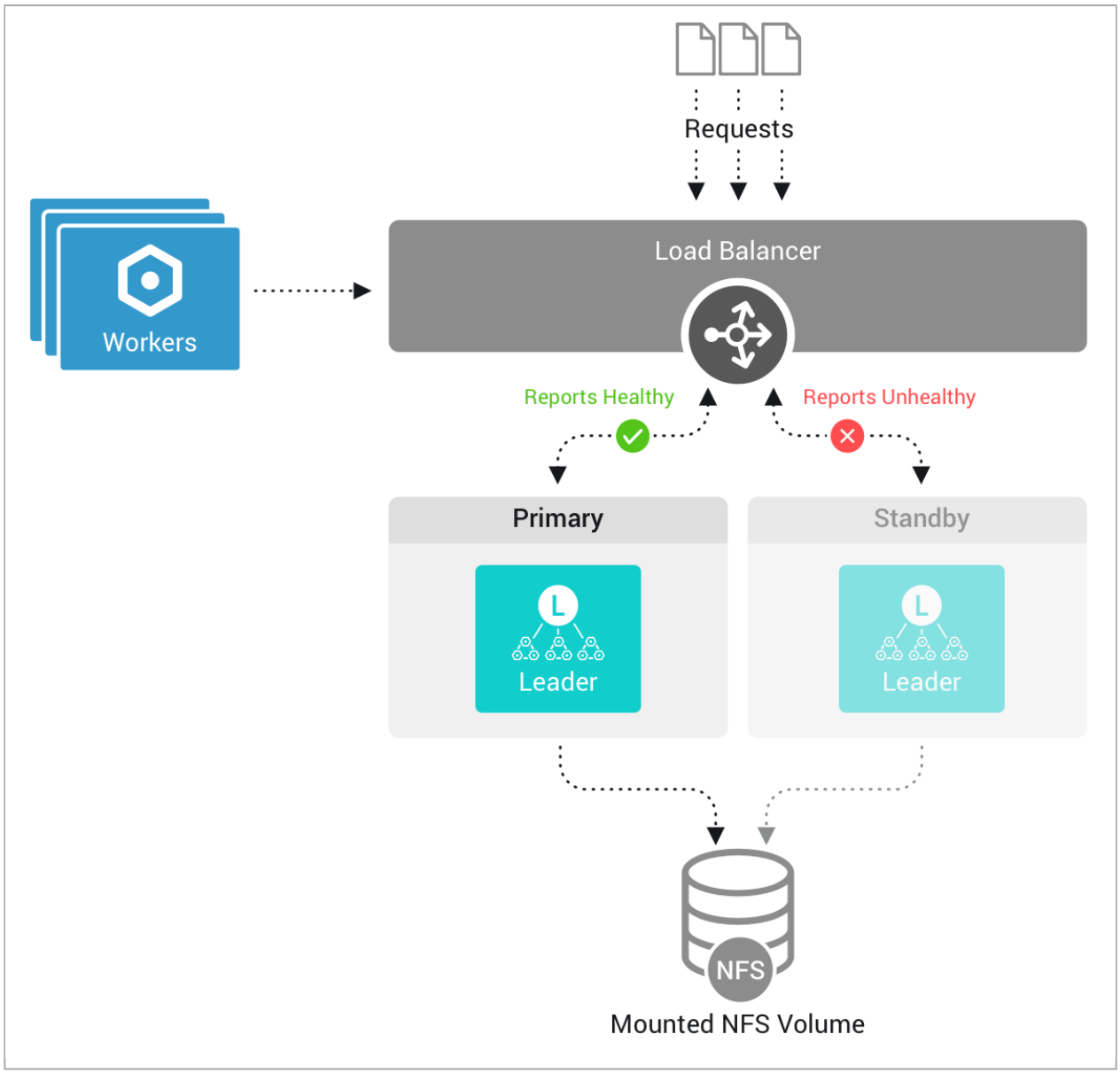
> Configuring a backup Leader requires a Cribl Stream Enterprise or Standard license.

How It Works

When you configure a second Leader, there will be only one active Leader Node at a time. The second Leader Node will be used only for failover. For this architecture to work, you must configure all failover Leaders' volumes to point at same the Network File System (NFS) volume/shared drive.

If the primary Leader Node goes down:

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



Leader High Availability/Failover Design

Required Configuration

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

Auth Tokens

Make sure that both Leaders have matching auth tokens. If you configure a custom **Auth token**, match its value on the opposite Leader.

- In the UI, check and match these values at each Leader's **Settings** > **Global Settings** > **Distributed Settings** > **Leader Settings** > **Auth token**.

- Or, from the filesystem, check and match all Leaders' instance.yml > `master` section > `authToken` values.

(If tokens don't match, Edge Nodes will fail to authenticate to the alternate Leader when it becomes active.)

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 via `/api/v1/health endpoint`.

The following load balancers support health checks:

- Amazon Web Services (AWS) Network Load Balancer (NLB).
- HAProxy.
- NGINX Plus.

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`.

Configuring 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

Using the UI

1. In **Settings** > **Global Settings** > **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 (e.g., `/mnt/cribl` or `/opt/cribl-ha`). Specify an NFS directory outside of `$CRIBL_HOME`. A 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 refreshes 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. Click **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.

Using 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.

Using 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.

Using Environment Variables

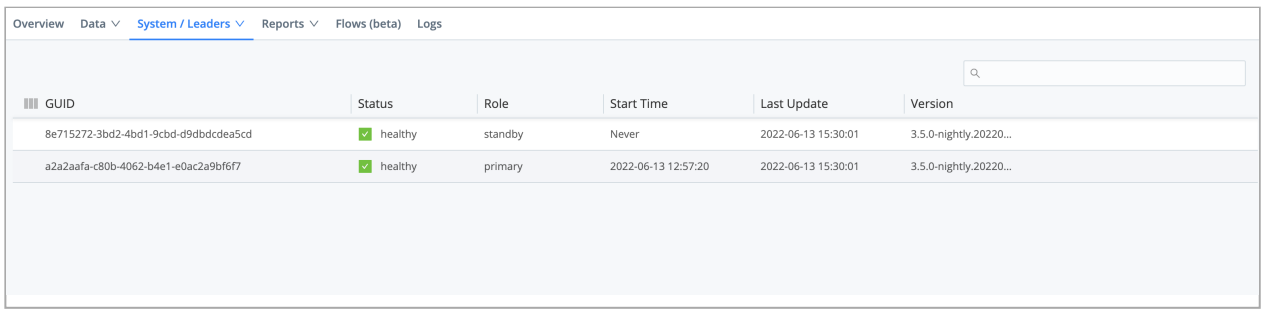
You can configure additional Leader Nodes via the following environment variables:

- `CRIBL_DIST_MASTER_RESILIENCY=failover`: Sets the Leader's `Resiliency` to `Failover` mode.
- `CRIBL_DIST_MASTER_FAILOVER_VOLUME=<shared_dir>`: Sets the location (e.g., `/mnt/cribl`) of the NFS directory to support Leader failover.
- `CRIBL_DIST_MASTER_FAILOVER_MISSED_HB_LIMIT`: Determines how many Lease refresh periods elapse before the 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.

Monitoring the Leader Nodes

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



Upgrading

When upgrading:

- Stop both Leaders.
- Upgrade (Stream, Edge) the Primary Leader, Second Leader, and then each Edge Node, respectively.

Disabling the Second Leader

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

5. Cribl Edge Better Practices

5.1. Monitoring your Infrastructure with Cribl Edge

Cribl Edge is designed to help 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, while there are a number of visualization tools providing charts, graphs, and alerts for the web when connected to supported data sources. Enhance the performance of your environments by combining Cribl Edge with your data visualization tool of choice, so you can identify and investigate issues and monitor applications faster than ever before.

This setup guide uses Grafana as an example visualization tool. We will walk you through the following:

- Configure your Cribl Edge Sources: Depending on your endpoints, you can use the System Metrics Source for Linux, or Windows Metrics, or Kubernetes Metrics.
- Configure the Grafana Cloud Destination to send data to Grafana Cloud.
- Access Grafana Cloud to see the metrics and visualizations.

Cribl has configured a Prometheus dashboard to display system metrics, and we've shared the dashboard in the Grafana library, which is public. This is a relatively simple dashboard, suitable for showing basic aggregate metrics.

Another useful dashboard for displaying Linux metrics is the one that's commonly used with Prometheus and its Node Exporter agent, see the Node Exporter Full dashboard. This dashboard can handle the highly detailed metrics. Grafana also provides a number of Windows dashboards that you can use to display Windows metrics.

Configure the System Metrics Source on Cribl Edge

In Cribl Edge, start by configuring and enabling the applicable Source for system metrics:

- Linux System Metrics Source
- Windows Metrics
- Kubernetes Metrics

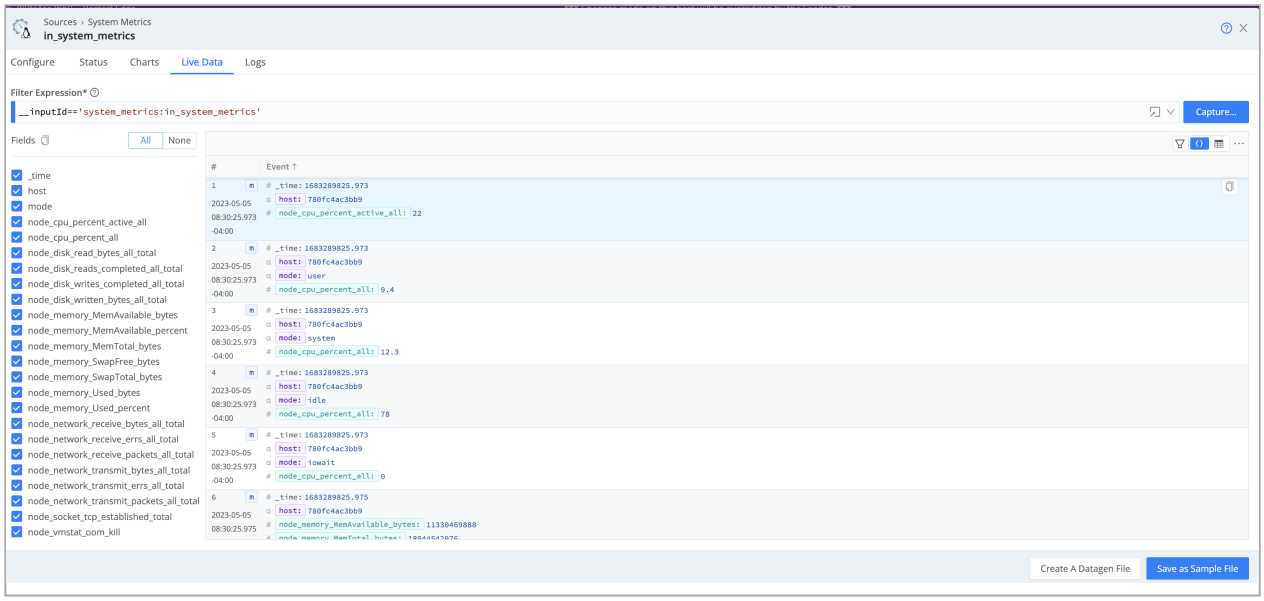
The key requirement here is to to decide on the level of granularity for the metrics you want to collect. In the System Metrics Source and Windows Metrics Sources, go to the **Host Metrics** and **Container Metrics** tabs:

- 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.

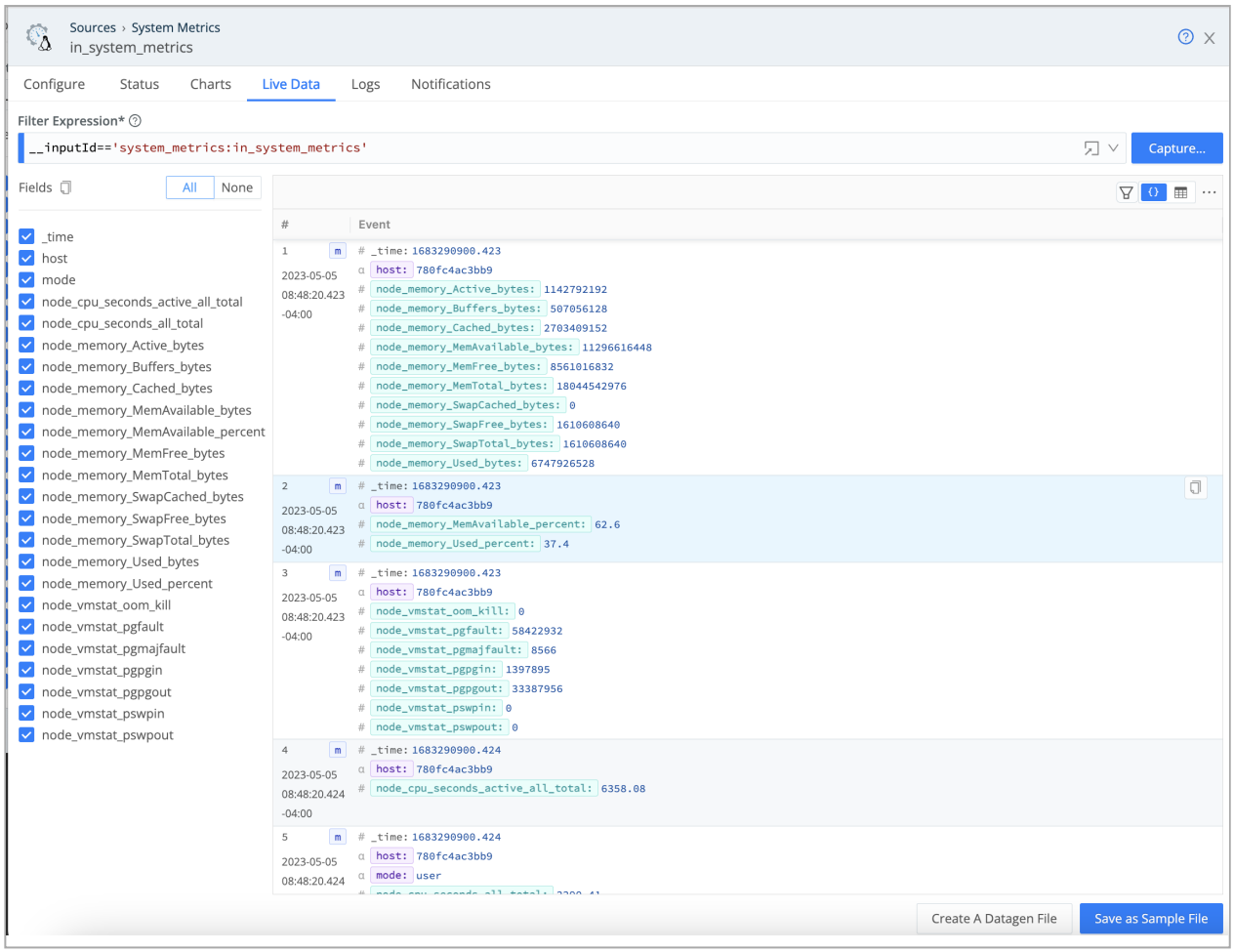
For the Kubernetes Metrics Source, use the **Filter Rules** to specify the Kubernetes objects that the Source should parse to generate metrics. If you specify no restrictive rules here, the Source will emit all events.

For all of the Sources, in the **Pre-Processing** tab, select the `prometheus_metrics` **Pipeline**.

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.



Basis System Metrics

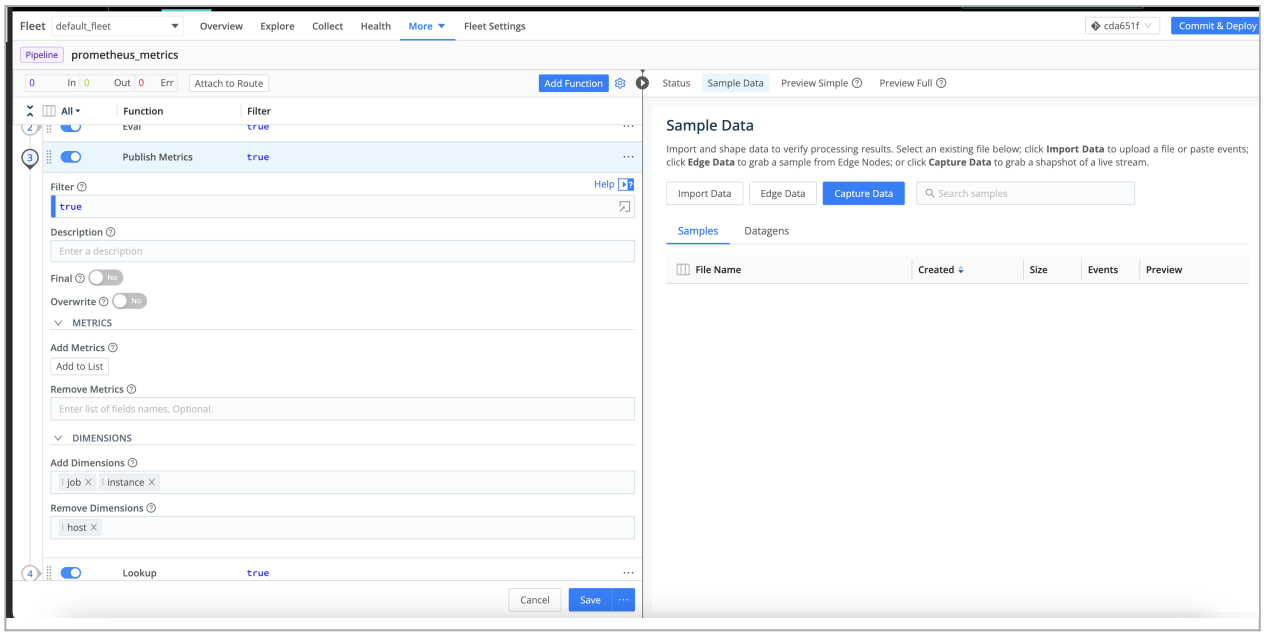


Detailed System Metrics

(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.

Select **Manage** from the top nav, then select a **Fleet** to configure. From a Fleet's top nav, select **More** > **Pipelines**.

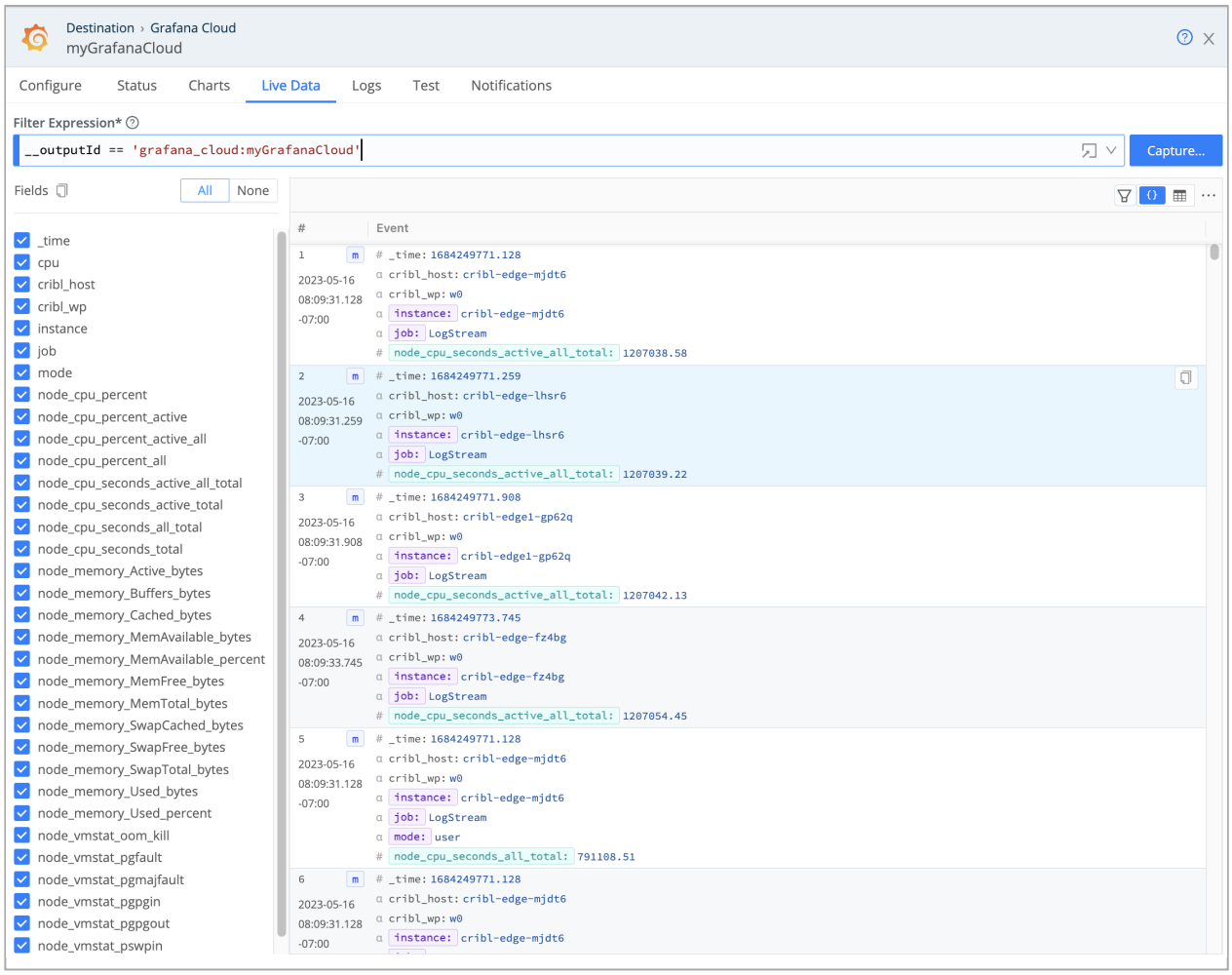


Customizing the Pipeline's dimensions

Configure the Grafana Cloud Destination

Next, we'll configure the Grafana Cloud Destination to send data to Grafana Cloud.

Preview your data on the Grafana Cloud config modal's **Live Data** tab.



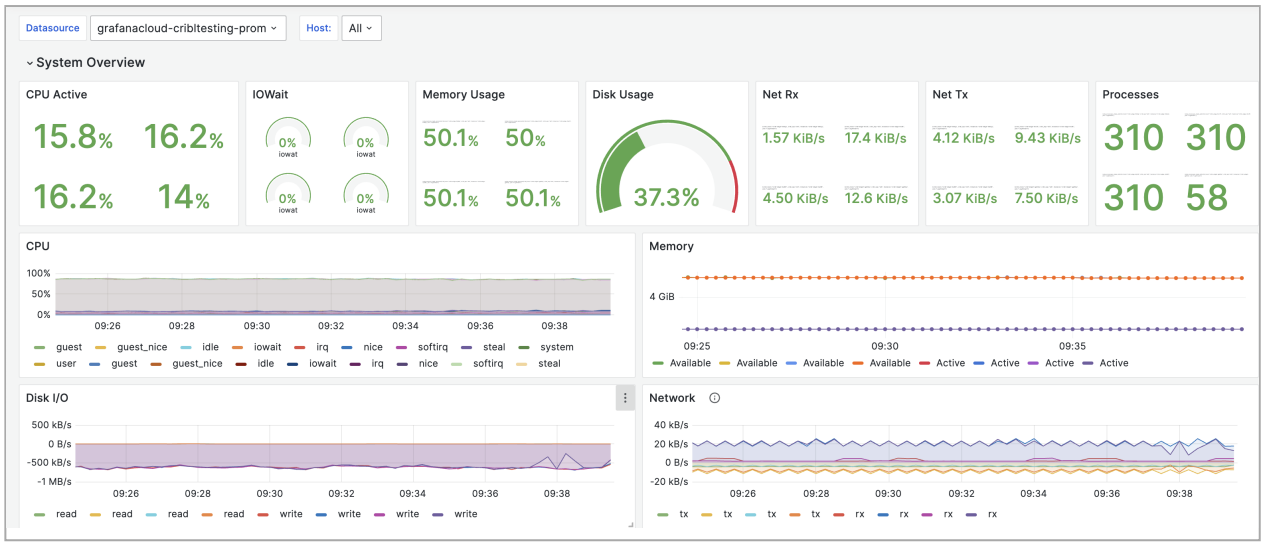
Preview Live Data

Visualize the Data in Grafana

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.

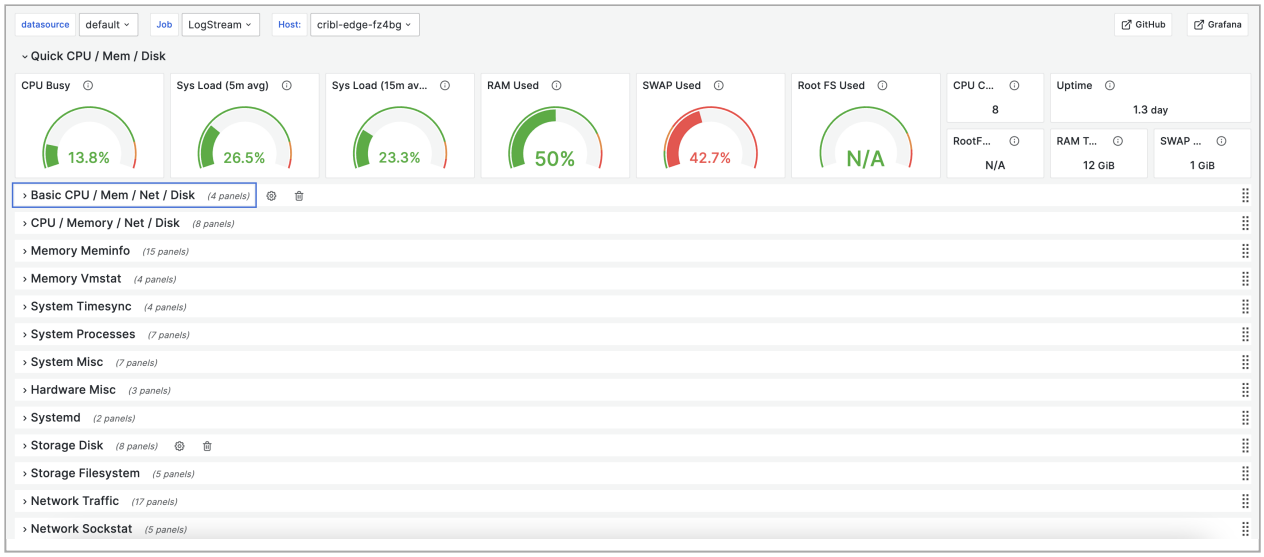
Use Grafana's **Filter** option to browse the dashboards for your Prometheus data. Below are sample dashboards with the basic and detailed metrics shown.

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

5.2. 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:

- Do not exceed 3,000 Edge Nodes per Fleet hierarchy, which is a Fleet and all of its Subfleets. Exceeding this maximum causes issues when the Leader pushes a configuration bundle. For details, see Leader Scalability.
- 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`.
- And lastly, since upgrades occur at the Fleet level, consider the number of Edge Nodes the Leader will be upgrading at once.

Fleet Design Better Practices for Enterprises

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

- Do not use the `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 you are limited to the `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, etc.

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

> 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 will still be 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 Packs and Pipelines can’t be inherited by a Subfleet from a Fleet at this time. They need to be recreated / imported into child fleets.

5.3. 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

Configuring Fleets & Fleet Settings on Cribl Edge

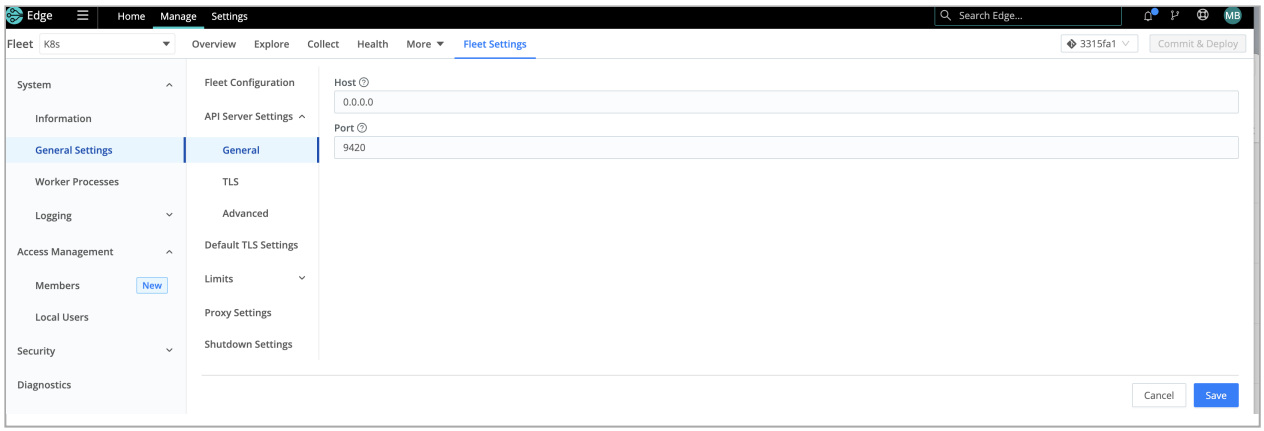
Before we deploy the Edge Node, let's configure a Fleet. Fleets and their corresponding Subfleets allow you to author and manage configuration settings for a particular set of Edge Nodes.

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.

Once you have configured your Fleet, you must adjust the API Server settings in Fleet Settings to bind to all IPs in the container - this is done by changing API Server configuration from `127.0.0.1` (Loopback Interface) to `0.0.0.0`. Kubernetes health checks will fail if this step is not performed prior to pushing config bundles to Edge agents. For details, see Health Checks.

To change the Fleet Settings:

1. Navigate to **Manage** in the Cribl Edge UI.
2. Select the Kubernetes-specific Fleet.
3. Click **Fleet Settings** in the submenu.
4. Navigate to **System** > **General Settings** > **API Server Settings** > **General**.
5. Change the **Host** value from `127.0.0.1` to `0.0.0.0`.
6. Click **Save**.



Fleet Settings

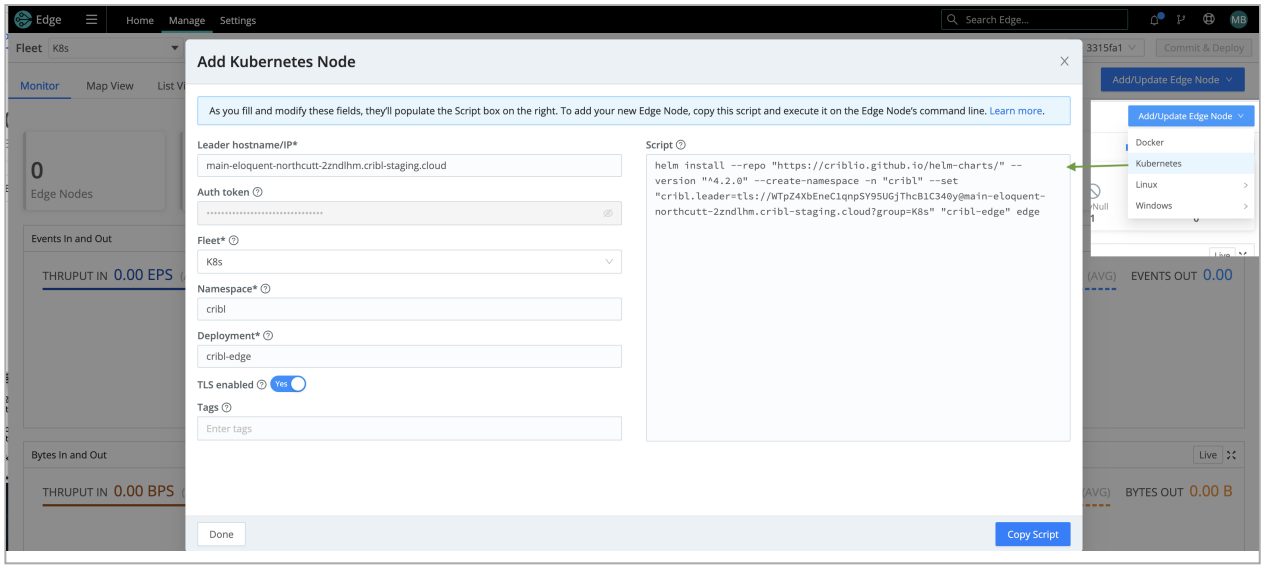
Use the Bootstrap Script to install the Helm Chart

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

Environment Variables

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

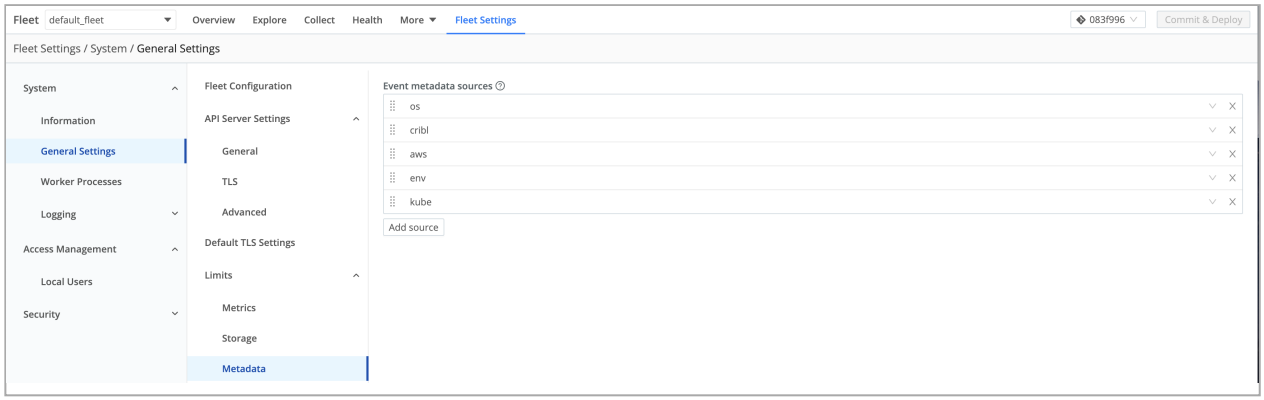
- `CRIBL_SERVICEACCOUNT_PATH` - by default, set to: `/var/run/secrets/kubernetes.io/serviceaccount`.
- `CRIBL_K8S_POD` - the name of the Kubernetes Pod in which Cribl Edge is deployed.
- `KUBECONFIG` - set this for local testing.
- `CRIBL_K8S_FOOTGUN` - set to `true` to enable resource-intensive, potentially risky modes of the Kubernetes Metrics and Kubernetes Logs Sources.
- `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.

Configuring 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.

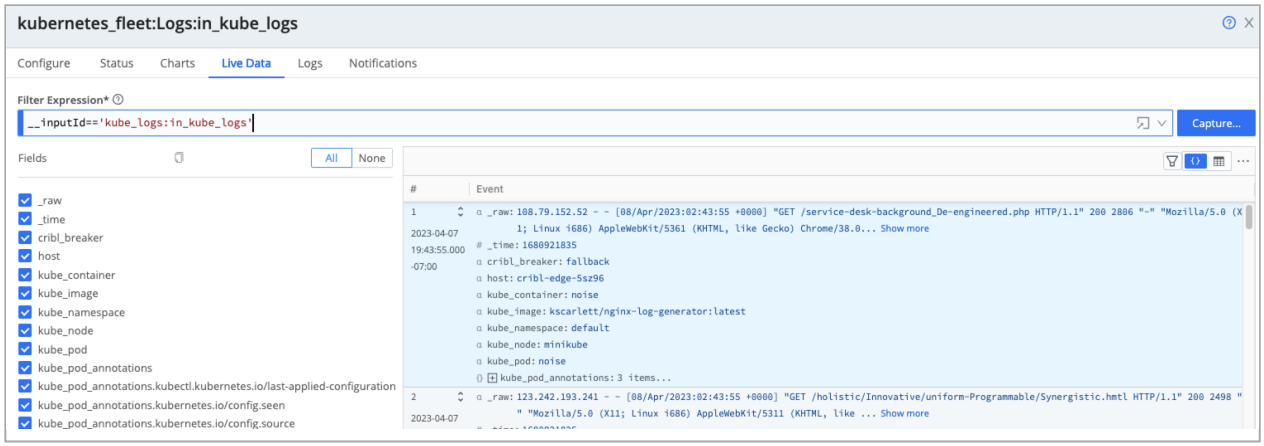
For example, if the container emits `1 MBps` and rotates logs at `10 MB`, the polling interval must be less than `10s`, to prevent hitting the `10 MB` rotation threshold twice in 10 seconds.

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'

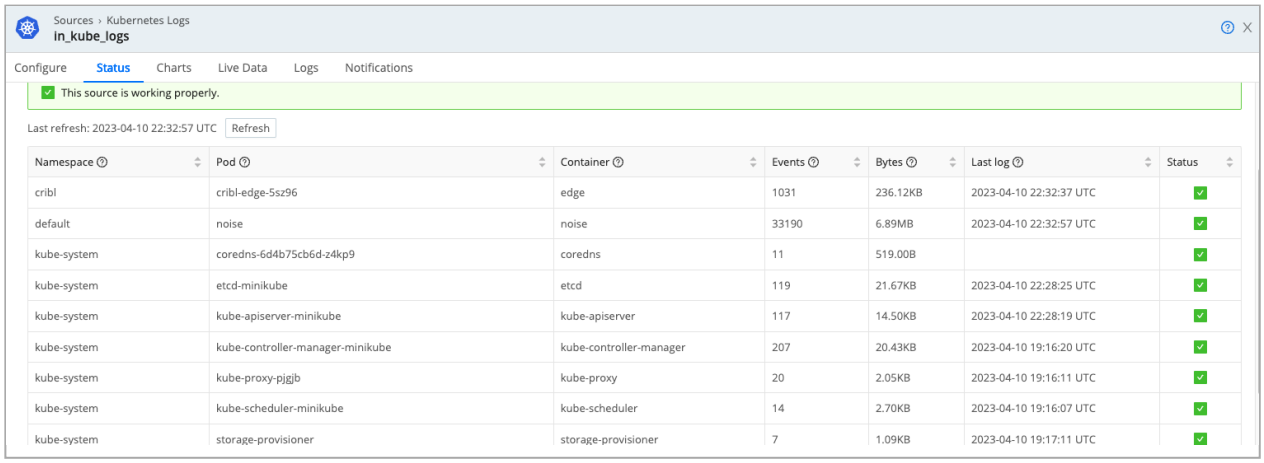
Configure the Kubernetes Logs Source to collect the Standard output (`stdout`) and Standard error (`stderr`) log streams from each Pod deployed inside the cluster.

Optionally, you can filter and enrich incoming logs.



Live Data: Kubernetes Logs Source

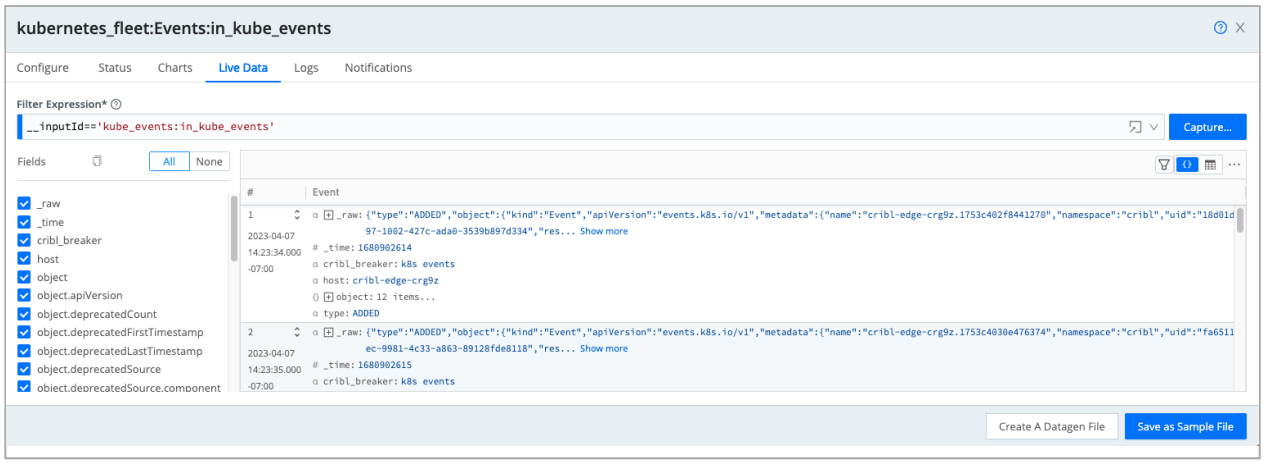
Below the **Capture** button, click the ellipsis (three dots or ...) and then enable the **Show Internal Fields** toggle to explore the `__metadata` fields forwarded from the Cribl Edge Nodes.These metadata fields show information about the Pod, Node, and Cribl environment where they were collected.



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.



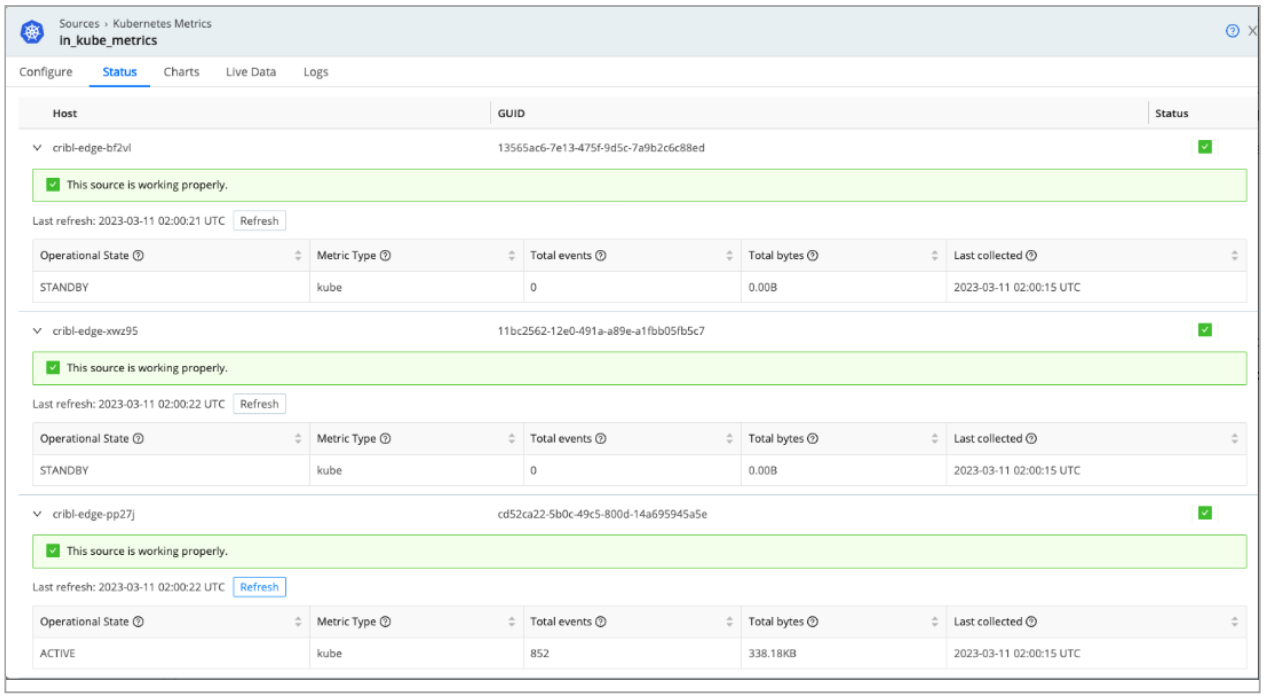
Live Data: Kubernetes Events Source

Kubernetes Metrics Source

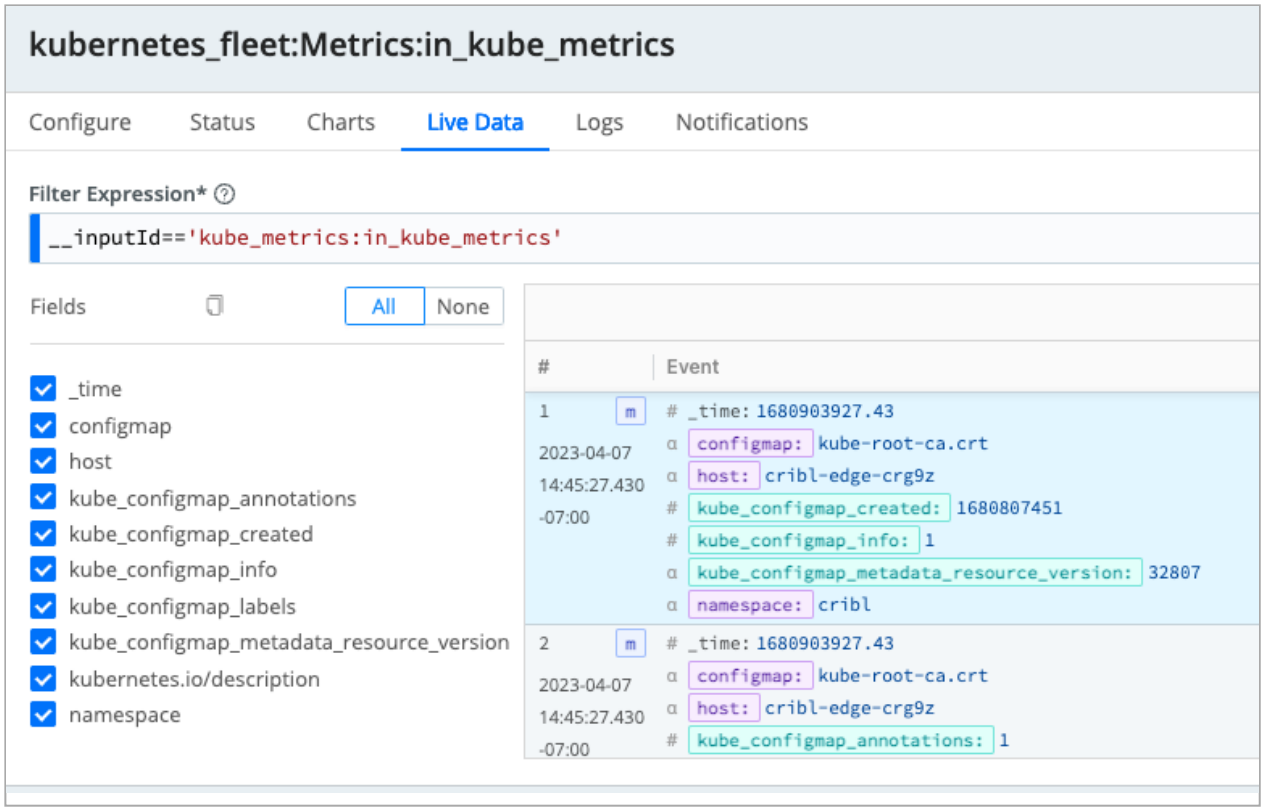
Configure the Kubernetes Metrics Source to generate metrics periodically based on the status and configuration of a Kubernetes cluster and its Nodes, Pods, and containers.

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

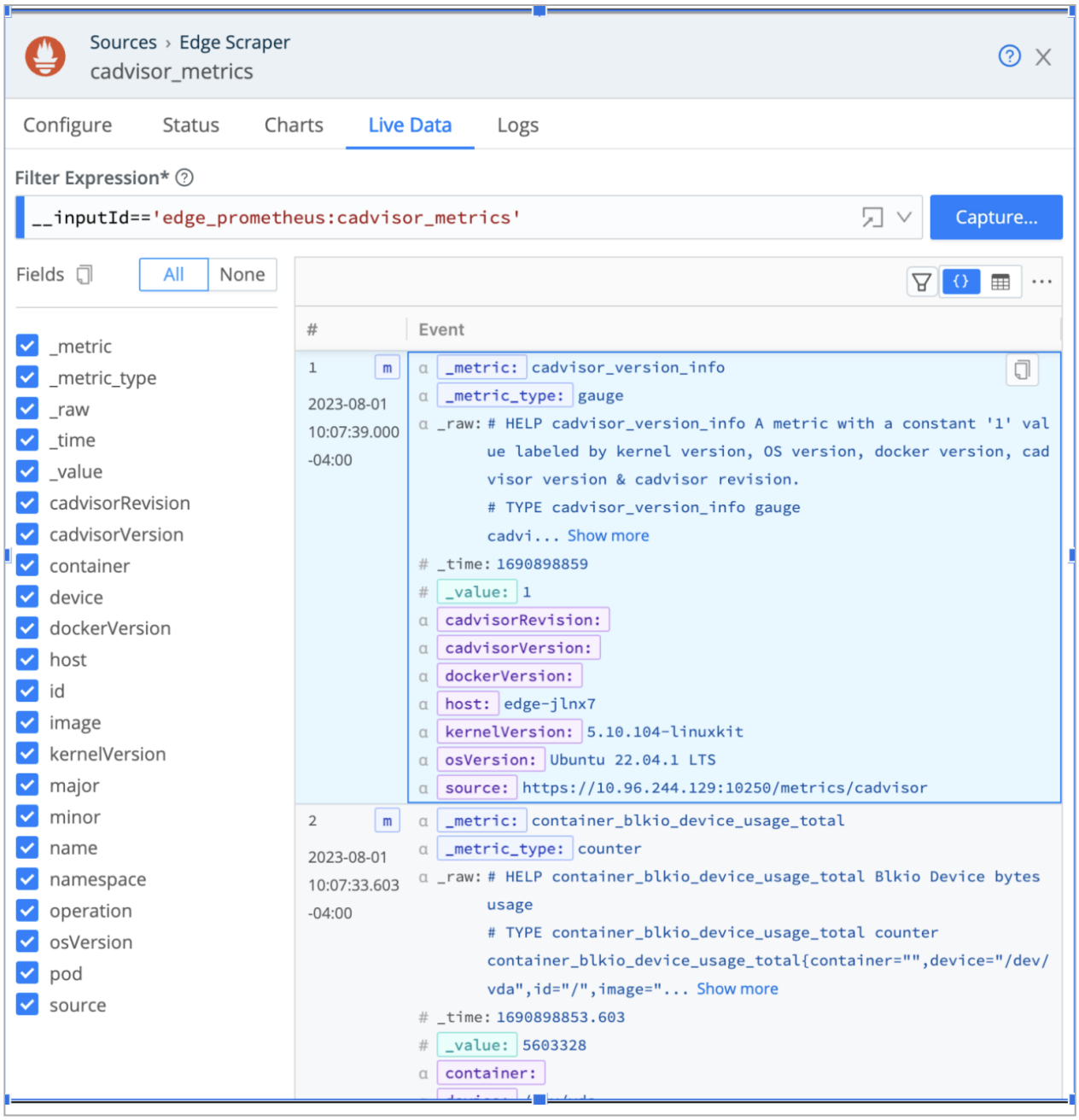
Prometheus Edge Scraper

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, go to **Manage** > **Collect** to attach the Kubernetes Logs Source to the Loki Destination, using the configured Pipeline as the connection.

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

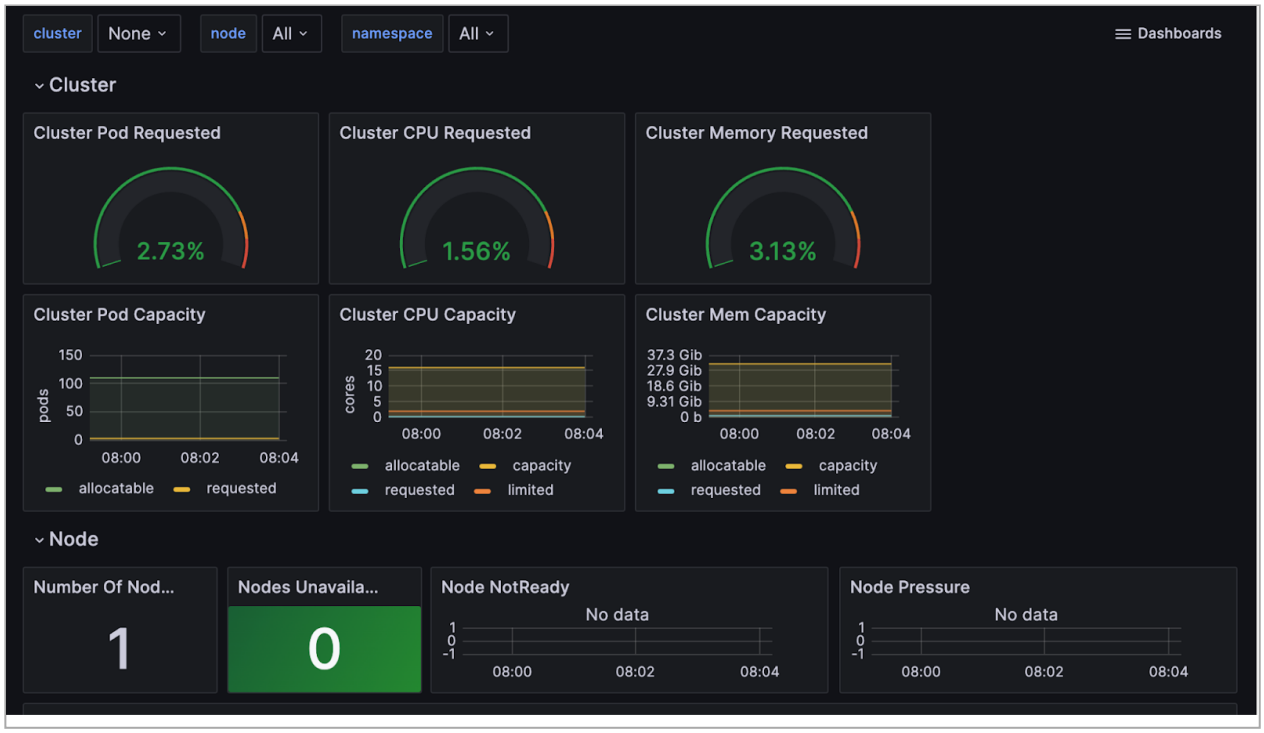
Next, go to **Manage** > **Collect** to attach the Kubernetes Metrics Source (or Prometheus Edge Scraper) to the Prometheus Destination, using the `prometheus_metrics1` Pipeline as the connection.

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.



5.4. 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 your Windows servers (2016, 2019, or 2022) to collect observability data (both metrics and logs) and send them to your desired destinations. By combining Cribl Edge with your favorite system 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. We will also cover how to create a Windows-specific Fleet to connect to your Cribl Leader and Mapping Rulesets to switch Fleets.
- Explore the Windows node and get a snapshot of the host system's current sate.
- 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.

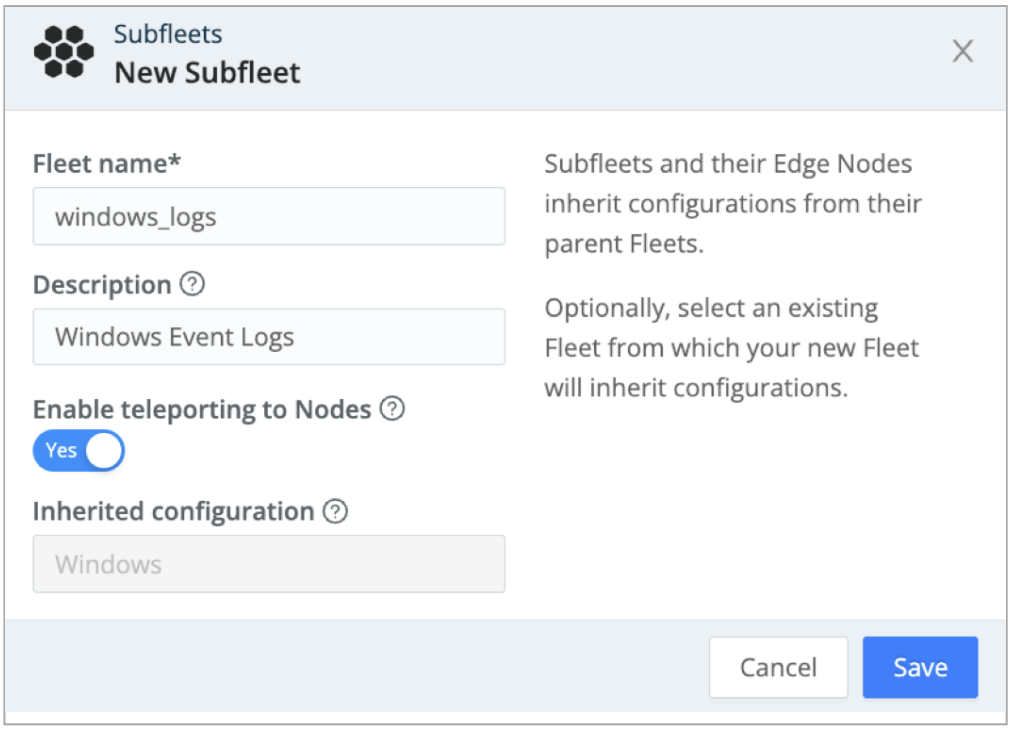
Deploy Cribl Edge on Windows

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`.



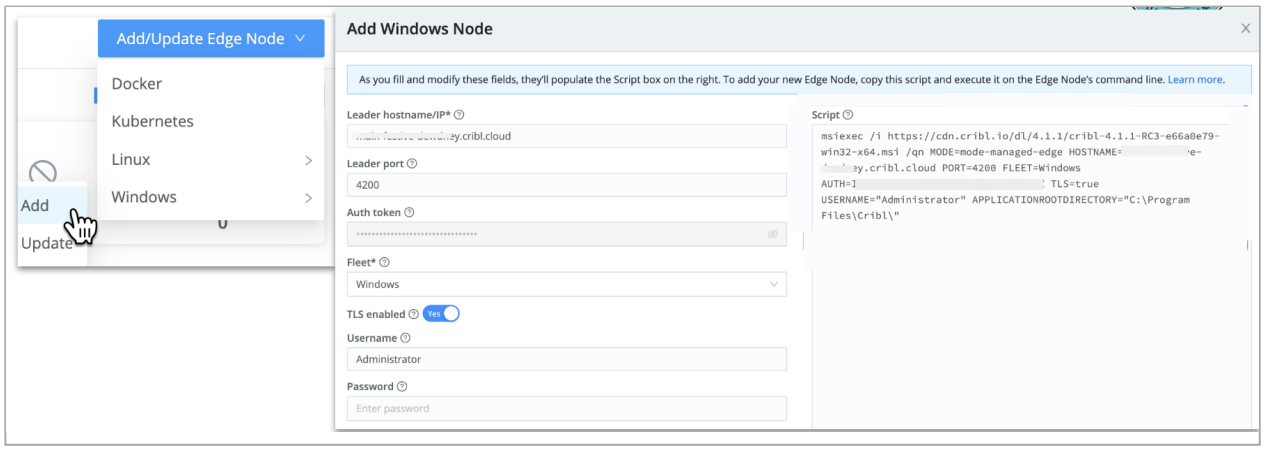
Create a Subfleet

> 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.

Use the Bootstrap Script to Add an Edge Node

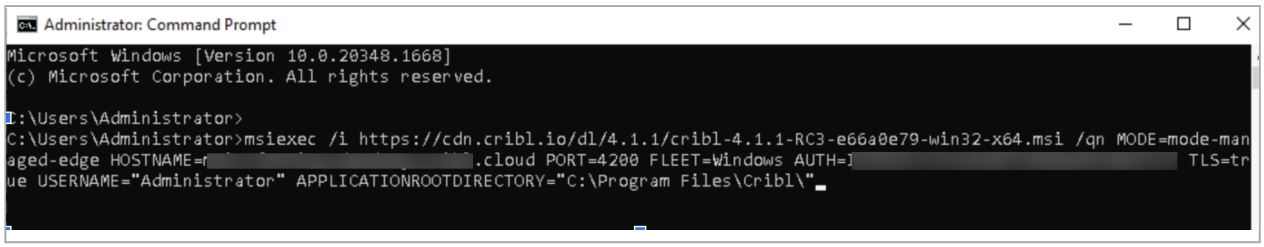
Next, bootstrap a new Windows Edge Node on your Fleet.

Below is a composite screenshot of adding an Edge Node on Windows.



Generating a script to bootstrap an Edge Node on Windows

To add the Edge Node, paste the script into your Windows command prompt and execute it. You must run the command line as an administrator for this to work. For details, see Start Command Prompt as an Administrator.



Pasting the script to the Command Prompt

You can edit the `instance.yml` files, to change the following:

- Reassign the Edge Node to a different Fleet.
- Enable TLS after installation for prior versions of Cribl Edge. As of Cribl Edge 4.1, this is included in the bootstrap script when you deploy from a Cribl.Cloud Leader. For details, see Enabling TLS After Installation.

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

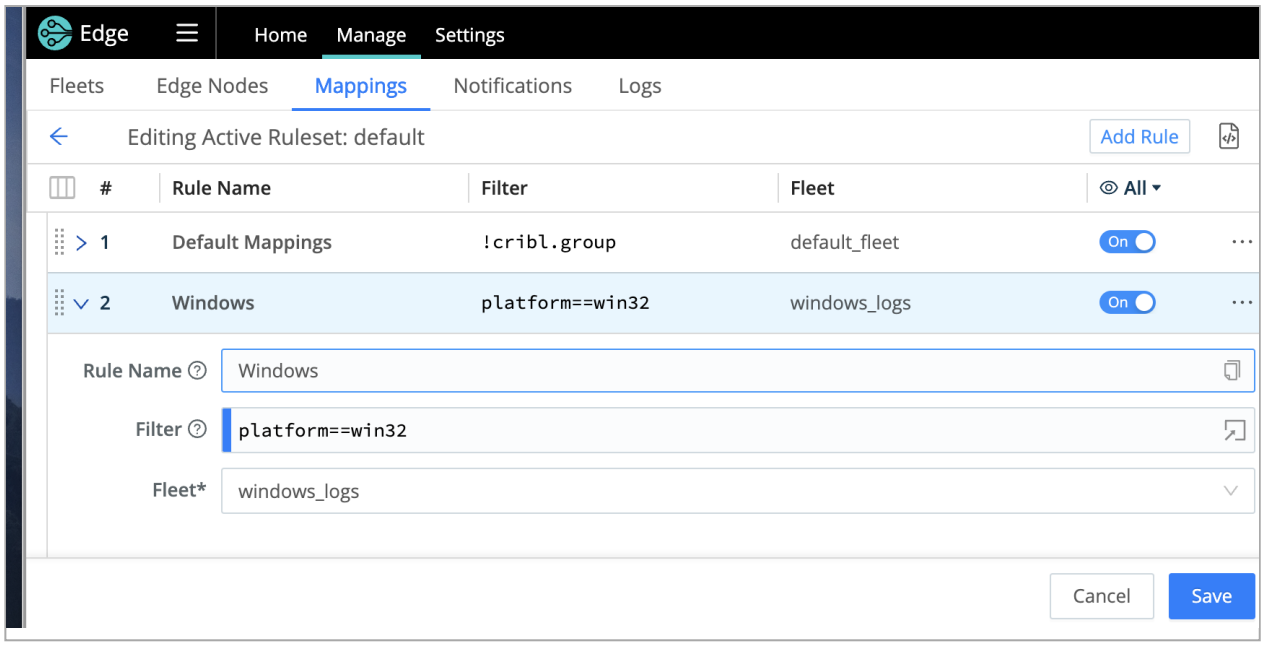
Keep in mind, there are two more changes related to deployment in Cribl Edge 4.1:

- The 4.1 MSI installer uses the `CRIBL_VOLUME_DIR`, so logs, configs, state, etc. are in `ProgramData\Cribl` instead of `Program Files\Cribl`. The `instance.yml` is located in ProgramData\Cribl too.
- The 4.1 MSI installer's command-line syntax changed, so if you simply change the MSI file it uses and reuse the earlier installation command line, you will get default settings, i.e., standalone Edge. The current syntax can be found in the Add Node UI.

Use a Mapping Ruleset to Switch Fleets

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

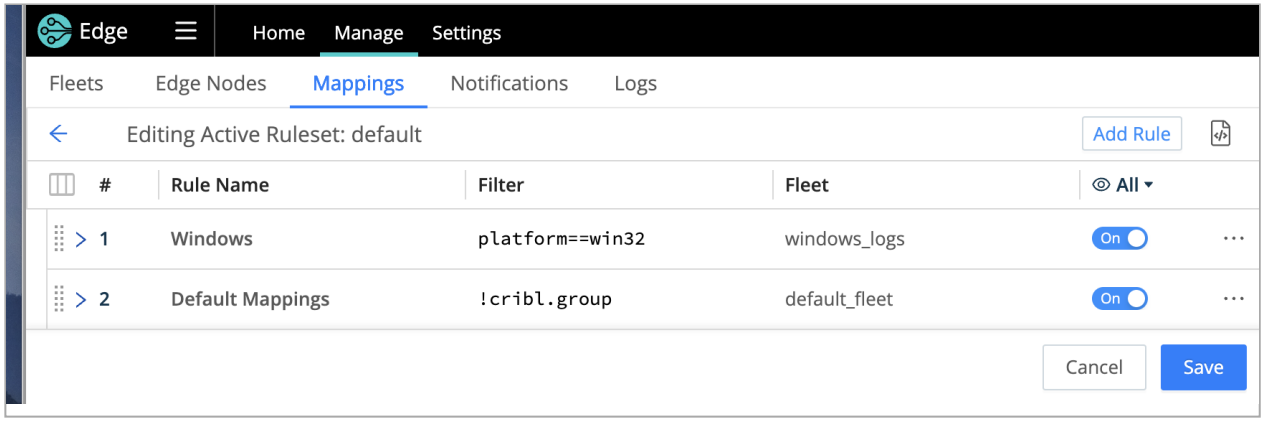
First, we'll add a new rule in the existing `default` ruleset.



Map to the SubFleet

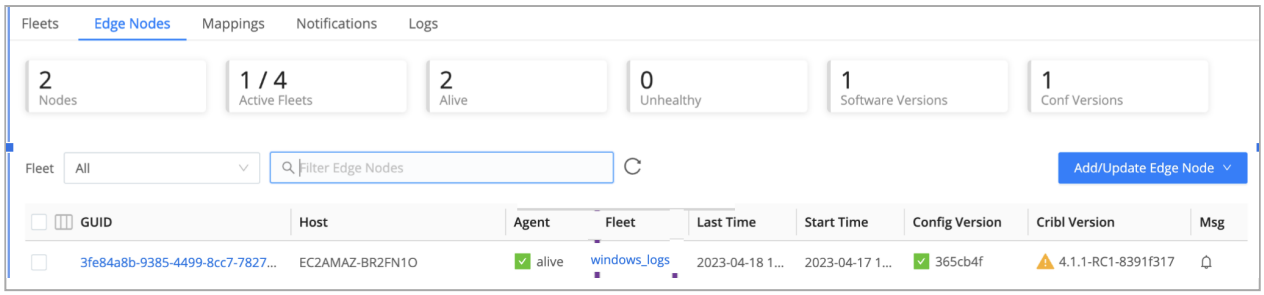
Only one Mapping Ruleset can be active at a time, although a ruleset can contain multiple rules. It's important to keep your Fleet assignments within one ruleset. This example uses the `default` ruleset.

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.

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 Exploring Cribl Edge on Windows.

From the Fleet's **Explore** menu, click 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

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.

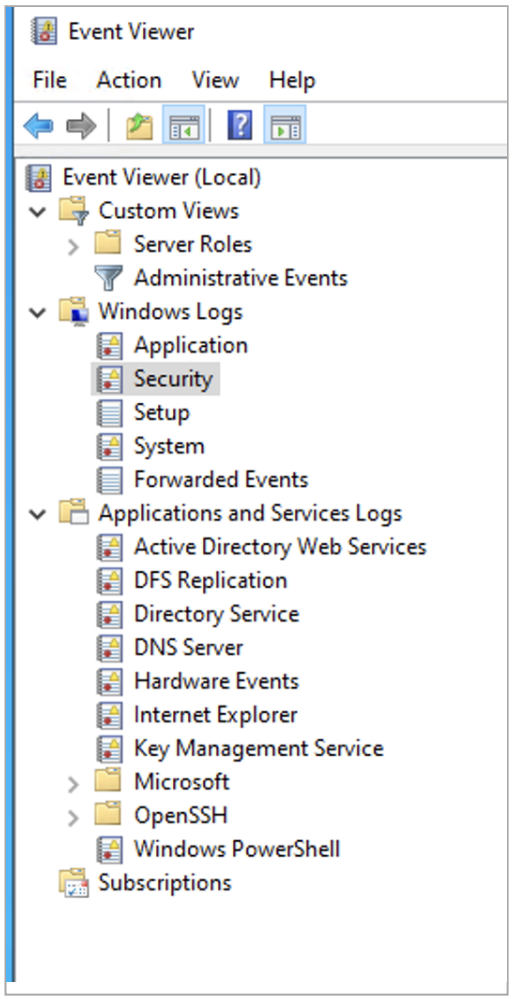
Event logs in Windows Servers are classified into the following broad categories:

- **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.

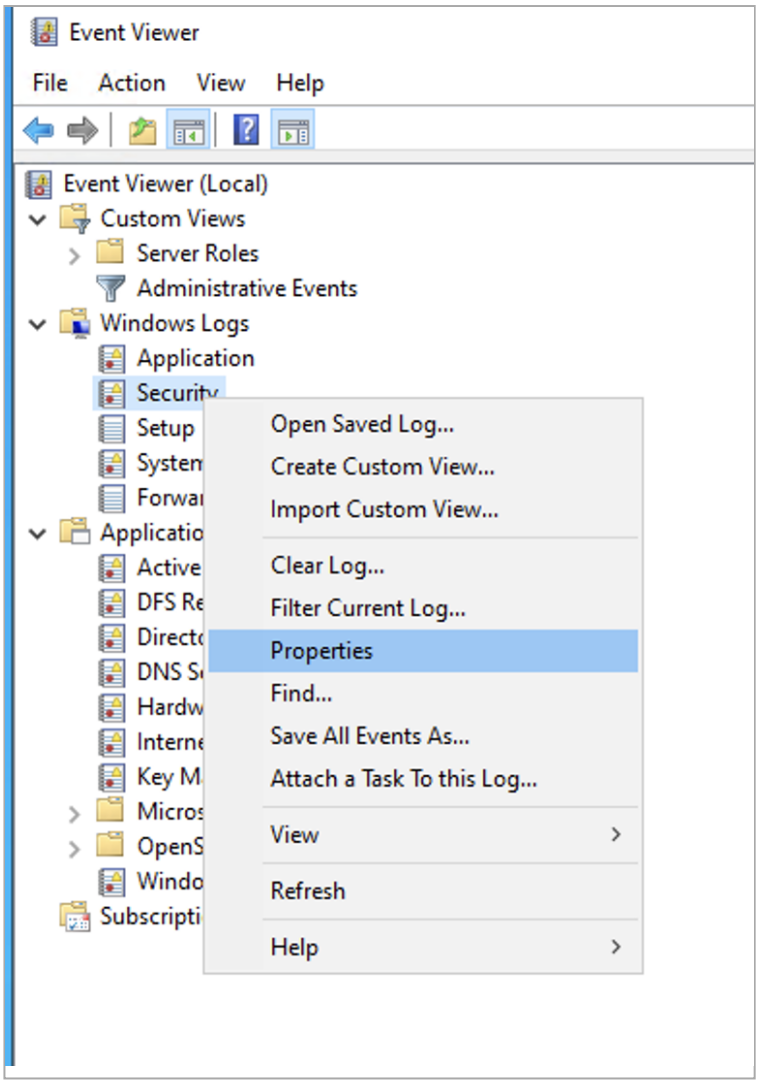
> There's a Known Issue registered for forwarded events. See Cascading problems when Windows Event Logs Source collects from ForwardedEvents channel for details.

To find out what other event logs might be of interest, go to your Windows Server Event Viewer.

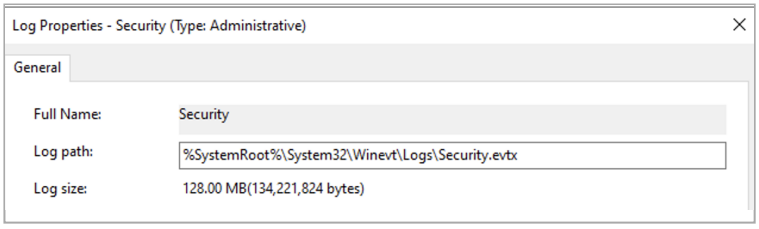


Event Viewer

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



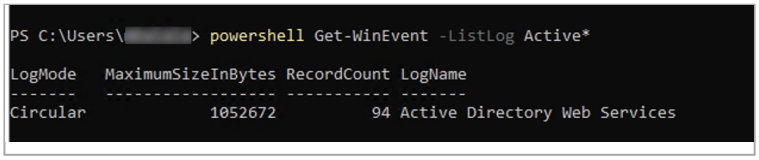
Log properties



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 the * as a shortcut.

```
powershell Get-WinEvent -ListLog Hardware*
```

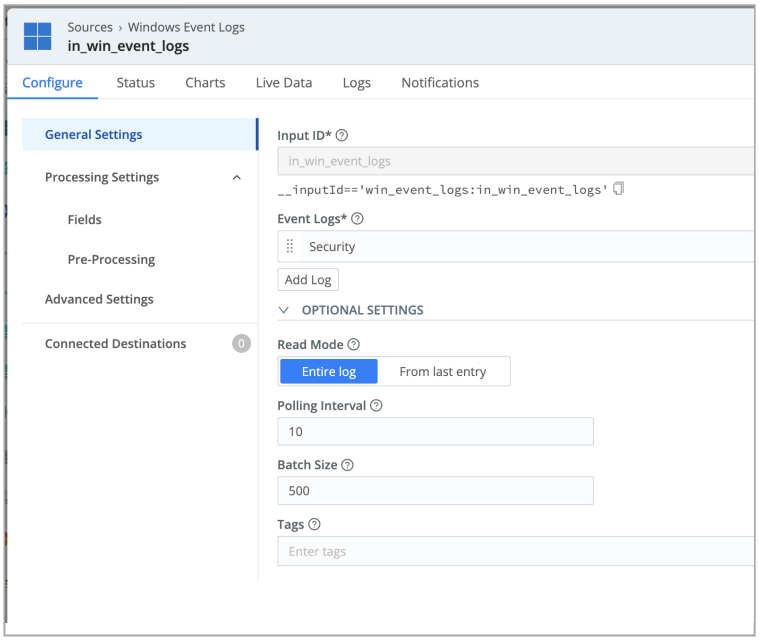


Log Name from PowerShell

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.

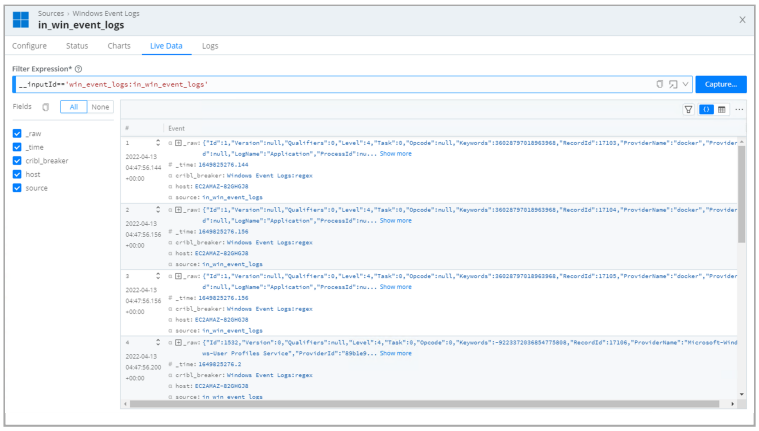
Configure the Windows Event Logs Source

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.



Configuring Windows Event Logs

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

In Cribl Edge, start by configuring and enabling the 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** tab, and select the following options for each type:

- **CPU**: Set to **Custom**, enable **Per CPU** metrics.
- **Network**: Set to **Custom**, enable **Per Interface** metrics.
- **Disk**: Set to **Custom**, enable **Per Volume** metrics.

On the **Pre-Processing** tab, select the `prometheus_metrics` Pipeline.

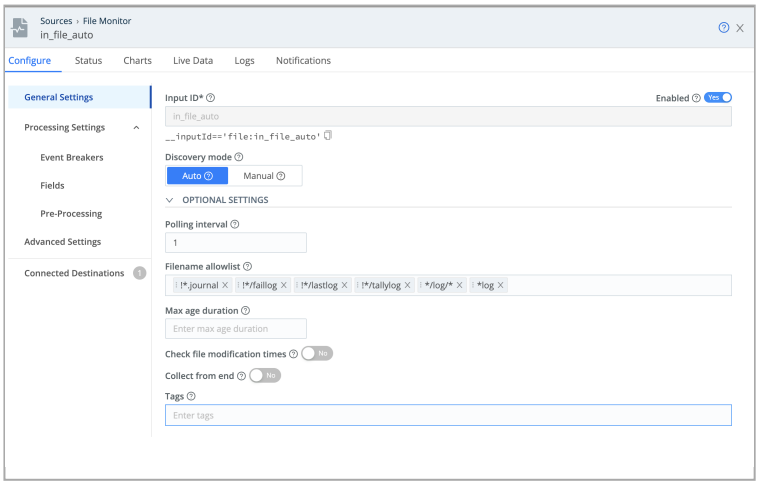
When done, **Commit** and **Deploy** your changes. Before moving on to the next step, check the **Live Data** tab to make sure the metrics are generated.

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

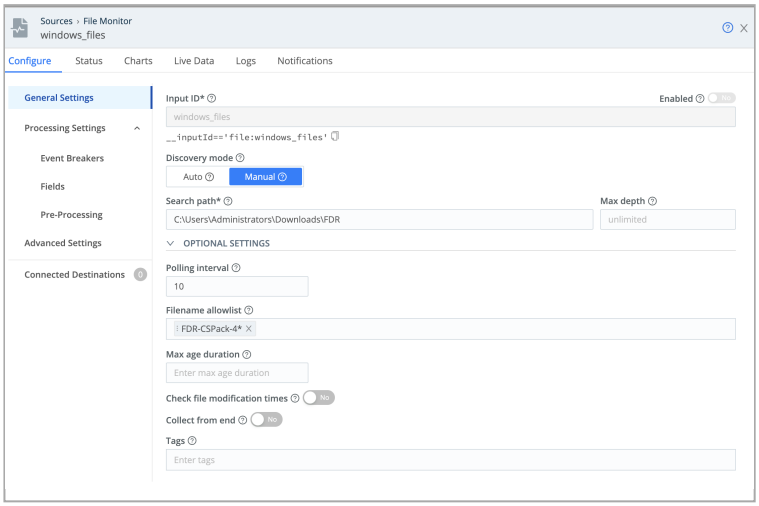
In Cribl Edge, start by configuring and enabling the File Monitor Source.

This configuration monitors all files ending with `.log`.



File Monitor in Auto mode

The **Manual** configuration option allows you to monitor a select file within a directory. When creating a new entry, specify the folder where the log files can be found. If the **Max depth** is empty, all subdirectories will be searched. To limit it to the referenced directory only, set **Max depth** to `1`. 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.



File Monitor Configuration in Manual mode

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 Collectors or other Sources.

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

> 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`

POWERSHELL PROMPT COMMAND	CRIBL EXEC SOURCE CONFIGURATION
`# 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:

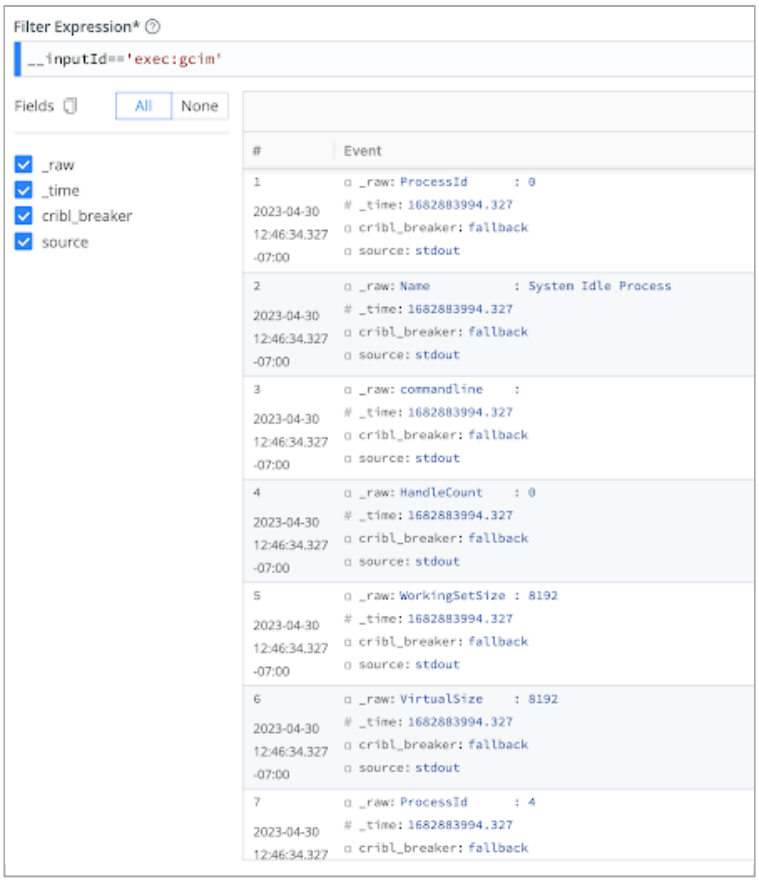
```
powershell -command "Get-CimInstance win32_process | select ProcessId, Name, commandline, HandleCount, WorkingSetSize,VirtualSize"
```

And for the newer version of the Powershell command, you can enter the following into the Exec Source's **Command** field:

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

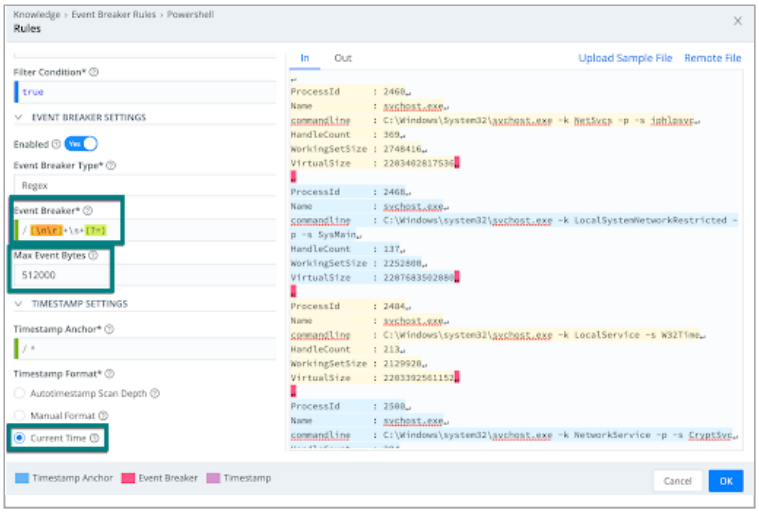
Exec Source's Command

The output of the command without using an Event Breaker:



Default Output

The output of the command with an Event Breaker:



Output with an Event Breaker

6. Administering

6.1. Upgrading

Teleporting an Edge installation into the future

Upgrading

This page outlines how to upgrade a Cribl Edge single-instance or distributed deployment along one of the following supported upgrade paths, which apply to UI-based upgrades:

CURRENT VERSION	UPGRADE PATH
4.x	4.x
3.x	3.x through 4.x
2.x	2.x through 4.x
1.7.x or 2.0.x	2.x.x, then 3.x or 4.x
1.6.x or below	1.7.x, then 2.x.x, then 3.x or 4.x

> If you're upgrading Edge Nodes directly from the command-line interface, you don't have to worry about a version upgrade path - there are no restrictions on the versions you can upgrade to or from.

You can upgrade Cribl Edge in two ways:

- Upgrade and Rollbacks via the UI
- Manual Upgrades and Rollbacks allows you to upgrade by uncompressing the new version on top of the old one.

Considerations

Here are some things to consider before you upgrade:

- 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.
- Before upgrading your Leader to v.4.0 and later, see Persisting Socket Connections, to prepare the host to keep communications open from Workers.
- Cribl Edge 4.1 and later encrypt TLS certificate private key files when you add or modify them. See instructions just below for backing up your keys from earlier versions.
- In Kubernetes, an Edge Node that is upgraded via the Leader reverts to its helm chart image after restarting a pod. There might be a version mismatch with the Leader as a result. Use the Helm Charts to upgrade the Edge Nodes on Kuberbenetes: `helm upgrade --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.

If you're planning to upgrade to v.3.5.4, note that any Edge Nodes must be at the same version as the Leader for v.3.5.4. Leaders running v.3.5.4 and later should test whether Edge Nodes are running a compatible version before deploying configs that could break Edge Nodes' data flow. The Leader will prompt you to upgrade these Nodes as needed.

To prevent Edge Nodes from failing on incompatible configurations, we introduced three new behaviors announced in Cribl Edge 3.5.4 release notes. These will make data throughput more resilient, but might enforce more-frequent (automated or explicit) upgrades to Workers' Edge versions:

- The Edge Leader will block configuration deployments to Edge Nodes running an earlier Edge version that's incompatible with the new configs. The Leader will prompt you to upgrade these Nodes to a compatible version.
- If Workers detect a Source or Destination type for which they have no supporting configuration, they will ignore (skip initializing) that integration, rather than failing.
- For unsupported Destination types, Workers will create a temporary placeholder Destination, but will exert Blocking backpressure until the configuration is updated. You will see a warning of the form: `Skipping configuring Destination...due to unrecognized type.`

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.

Upgrade and Rollback via the UI

> Currently, you can't upgrade **Windows** Edge Nodes via Cribl Edge's UI. You must re-run a Windows installer on these Nodes to upgrade them. This Windows limitation also applies to the Rollback section below.
>
> You can use the **Add/Update Edge Nodes** option in the UI, to generate an upgrade script for Windows. For details, see Add/Update Edge Nodes.

To upgrade via the UI, go to **Settings** > **Global Settings** > **System** > **General Settings** > **Upgrade**. The following controls are available here:

Package Source

The default **CDN** button downloads a package directly from Cribl's content delivery network. Selecting the alternative **Path** button exposes these additional controls:

- **Package location**: Enter either a URL (HTTP) or a local path to the upgrade package. HTTP example: `https://cdn.cribl.io/dl/4.1.0/cribl-4.1.0-6979aea9-linux-x64.tgz`
- **Package hash location**: Enter either a URL (HTTP) or a local path to the hash that validates the package. Supports `sha256` and `md5` formats. You can simply append `.sha256` to the contents of the **Platform-specific package location** field. HTTP example: `https://cdn.cribl.io/dl/4.1.0/cribl-4.1.0-6979aea9-linux-x64.tgz.sha256`

- **Save**/**Cancel** buttons: Click **Save** to store the specified locations. Clicking **Cancel** restores the **CDN** package-source selection.

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

Automatic Upgrades

Disable automatic upgrades defaults to `Yes` in on-prem/customer-managed deployments, to prevent Edge from automatically upgrading out-of-date Worker Nodes.

The automatic upgrade process works like this:

1. The Leader pulls packages and checks their hashes.
    - The Leader must (1) be able to connect to the path, and (2) have privileges to download the files.
    - If the path is a file path, the Leader copies the files to a known location in its filesystem.
2. Edge Nodes pull packages and check their hashes.
    - Edge Nodes pull from the leader through HTTP, not directly from the Leader's filesystem.
    - Each Edge Nodes pulls the package that is appropriate for that Node's platform and architecture.
3. Edge Nodes install the packages.

Upgrade/Upgrade Leader

In a Single-instance deployment, the **Upgrade** button is the only other control provided. In a distributed deployment, you have an additional option to upgrade **Fleets**. (As with manual upgrades, always upgrade the Leader before upgrading the Edge Nodes.)

Upgrade Fleets

This second tab, displayed only in distributed deployments, shows each Fleet's status.

> Upgrading Edge Nodes from the Leader requires a Standard or Enterprise license.

Click any row's **Upgrade** button to upgrade that Fleet. The resulting **Upgrade Fleet** dialog offers two states: Basic Upgrade and Advanced Upgrade.

Basic Upgrade Configuration

In this default **Upgrade Fleet** dialog, you can simply upgrade the whole Fleet, by clicking the dialog's **Upgrade** button to confirm.

Cribl Edge will check to ensure that Edge Nodes are upgraded no higher than the Leader's version. Upgrades are performed as the user that was running Edge on each machine.

Advanced Upgrade Configuration

Click the dialog's gear () button to expose these additional options:

Quantity %: Specify what percentage of the Fleet's Edge Nodes to upgrade in this operation. If you enter a value less than the default `100`%, Edge will perform a partial upgrade, keeping the remaining Edge Nodes active to process data.

Rolling upgrade: This option upgrades Edge Nodes in batches, each sized according to your **Quantity %** value. When enabled, this slider also enables the dialog's two remaining controls:

Retry delay (ms): How many milliseconds to wait between upgrade attempts. Defaults to `1000` ms (1 second).

Retry count: How many times to retry a failed upgrade. Defaults to `5`.

After you click the **Upgrade** confirmation button, the **Fleet Upgrade** tab will display an additional button on this Fleet's row:

View: Click to display the upgrade task's status in the Job Inspector modal – select that modal's **System** tab to access details.

> When you initiate an upgrade via the UI, the new package is untarred to `$CRIBL_HOME/unpack.<random-hash>.tmp`. This location inherits the permissions you've already assigned to `$CRIBL_HOME`.

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 Worker 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`.

Manual Upgrades and Rollbacks

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 `/opt/`: `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 the `cribl/bin/cribl` directory if it's empty, and untar again. If you have **custom functions** in `cribl/bin/cribl`, please move them under `$CRIBL_HOME/local/cribl/functions/` before untarring again.

4. Restart Edge.

Distributed Deployment

For a distributed deployment, the general order of upgrade is: First upgrade the Leader Node, then upgrade the Edge Nodes, then commit and deploy the changes on the Leader.

> For distributed environments with a second Leader configured, the order of upgrade is: First upgrade the Primary Leader Node, then upgrade the Secondary Leader Node, then upgrade the Edge Nodes.

Upgrade the Leader Node

1. Commit and deploy your desired last version. (This will be your most recent checkpoint.)
    - Optionally, `git push` to your configured remote repo.
2. Stop Cribl Edge.
    - Optional but recommended: Back up the entire `$CRIBL_HOME` directory.
    - 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.
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 `/opt/`: `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 like those below will appear until the Edge Nodes are upgraded.

Upgrade the 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 `/opt/`: `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.

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 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.

> Cribl University offers a course titled Edge Administration that provides an illustrated overview of administering Cribl Edge. To follow the direct course link, first log into your Cribl University account.

(To create an account, click 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.

6.2. 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 – e.g., the `.dat` file and other local configs.

Uninstalling a Distributed Deployment

In a distributed deployment, repeat the above steps on:

- The Leader instance.
- All Edge Node instances.

6.3. 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
```

6.4. 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 volume. These licenses do not require sending telemetry metadata.

> This page does not apply to Cribl.Cloud plans. See Deploy Cloud for information about the Cribl.Cloud Enterprise plan.

Managing Licenses – Adding and Renewing

You can add and manage licenses at **Settings** > **Global Settings** > **Licensing**. Click **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.

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 Worker Group/Fleet managed by Leader Node A to a different Worker Group/Fleet managed by Leader Node B. In such situations, the same license used on Leader Node A can be used on Leader Node B.

Data generated by Cribl Internal Sources normally does 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:

- Data transferred between Cribl Stream Workers via the Cribl HTTP and Cribl TCP Sources does not count against your ingestion quota.
- Data sent from Cribl Edge's Cribl HTTP and Cribl TCP Destinations to Cribl Stream is counted against quota only in Cribl Edge.
- Data generated by Datagens does not count against your ingestion quota.

License Types

Cribl offers four Cribl Edge license types, summarized below.

> License terms are subject to change. For a detailed comparison of what's currently included in each license type, please see Cribl Pricing.

Enterprise License

This is a license available for purchase.

- Up to unlimited data ingestion.
- Up to unlimited Worker Groups and Fleets.
- Up to unlimited Worker Processes and Edge Nodes.
- Role-based access control.
- External authentication (via LDAP, Splunk, and OpenID Connect identity providers).
- Git remote backup.
- All other Cribl Edge features included.

Contact Cribl Sales at sales@cribl.io for more information.

Standard License

This is a license available for purchase. Compared to an Enterprise license, it offers a cost discount, in exchange for some limitations (all data volumes below based on uncompressed data size):

- Daily ingestion up to 5 TB/day.
- Maximum 1 Worker Group and 1 Fleet.
- Maximum 50 Worker Processes, and 100 Edge Nodes.
- External authentication supported, with undifferentiated Roles: all users are imported as `admin`.
- Git remote backup.

Contact Cribl Sales at sales@cribl.io for more information.

Free License

Free licenses ship in the download package, and are permanent. They impose some limitations:

- Daily ingestion up to 1 TB/day.
- Maximum 10 Worker Processes, and 100 Edge Nodes.
- Maximum 1 Worker Group and 1 Fleet.

The Cribl Edge Free license requires sending of anonymized telemetry metadata to Cribl. This license will block inputs if sending fails, after a grace period of 24 hours.

> **"One" License**
>
> As of February 1, 2022, Cribl simplified Free licensing by retiring the former "LogStream One" license. This license type, which carried a slightly different mix of limitations and privileges, will no longer be issued to new licensees.
>
> However, if you have a current LogStream One license, you can continue to use it, and renew it, for as long as you like. Contact your Cribl sales rep, or sales@cribl.io, to arrange your annual renewal.

Sales Trial License

A license type used when preparing a POC (proof of concept), or a pilot, whose requirements go beyond those afforded by the Free license. Contact Cribl Sales at sales@cribl.io for more information.

Combining License Types

Multiple license types can coexist on an instance. However, only a **single type** of license can be effective at any one time. When multiple types coexist, the following method of resolution is used:

- If there are any unexpired Enterprise or Standard licenses – use only these licenses to compute the effective license.
- Else, if there are any Sales Trial licenses – use only Sales Trial licenses to compute the effective license.
- Else, if there exists a Free license – use only the Free license to compute the effective license.

When an Enterprise or Standard license expires, Cribl Edge will fall back to the Sales Trial or Free type. However, an expired Sales Trial license cannot fall back to a Free license.

> **When Licenses Expire**
>
> 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/Edge Node Processes, Cribl Edge will attempt to balance (or rebalance) Worker/Edge Node (threads) as evenly as possible across all licensed Worker/Edge Nodes.

On Cribl Stream (LogStream) 2.3 or later, and Cribl Edge, you need to configure licensing only on the Leader Node. (See Managing Licenses – Adding and Renewing.) The Leader will push license information down to Worker Groups/Fleets as part of the heartbeat (Edge, Stream).

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/`.

Testing 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!
```

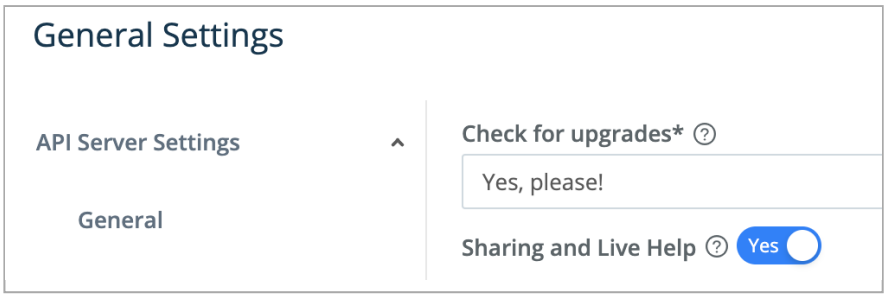
If you get a 302 response code, check whether you've omitted the URL's trailing `/`.

Disabling Telemetry and Live Help

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's UI at **Settings** > **Global Settings** > **System > General > Upgrade & Share Settings > Sharing and Live Help**. Toggle the slider to **No**.



Sharing and Live Help toggle

Metadata Shared Through Telemetry

Your Cribl Edge instance shares the following metadata with Cribl per interval (roughly, every minute):

- 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

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.

Licensing FAQ

How do I check my license type, restrictions, and/or expiration date?

Open Cribl Edge's **Settings** > **Global Settings** > **Licensing** page to see these details.

How can I track my actual data ingestion volume over the last 30 days?

Use the dashboard at **Monitoring** > **System** > **Licensing** to review 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`.

How does Cribl enforce license limits?

If your data throughput exceeds your license quota, Chuck Norris will track you down and make your life a living hell.

However, that will happen only in your nightmares. In the product itself:

- 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 License Types.

I'm using LogStream 2.3.0 or higher, with its "permanent, Free" license. Why is LogStream claiming an expired license, and blocking inputs?

This can happen if you've upgraded from a LogStream version below 2.3.0, in which you previously entered this earlier version's Free (time-limited) license key. To remedy this, go to **Settings** > **Global Settings** > **Licensing**, click to select and expand your expired Free license, and then click **Delete license**. Cribl Edge will fall back to the new, permanent Free license behavior, and will restore throughput.

If I pull data from compressed S3 buckets, is my license quota applied to the compressed or the uncompressed size of the file objects?

To measure license consumption, Cribl Edge uses the uncompressed size.

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, click 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.

6.5. 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 two randomly-selected endpoints simultaneously, 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.

6.6. Persistent Queues

Cribl Edge's persistent queuing (PQ) feature helps minimize data loss if a downstream receiver is unreachable. PQ provides durability by writing data to disk for the duration of the outage, and forwarding it upon recovery.

Persistent queues are implemented:

- On Push Sources.
- On Streaming Destinations. (Sources can also take advantage of a Destination's queue.)

Persistent Queues Supplement In-Memory Queues

Persistent queues trigger differently on the Destination versus Source side.

Destination Side

On each Cribl Edge Destination that supports PQ, an in-memory buffer helps the Destination absorb temporary imbalances between inbound and outbound data rates. E.g., if there is an inbound burst of data, the Destination will store events in the queue, and will then output them at the rate to which the receiver can sync (as opposed to blocking or dropping the events).

Only when this buffer is full will the Destination impose backpressure upstream. (This threshold varies per Destination type.) This is where persistent queues help safeguard your data.

Source Side

In Cribl Edge 3.4 and later, Push Sources' config modals also provide a PQ option. When enabled, you can choose between two trigger conditions: `Always On` mode will use PQ as a buffer for **all** events, whereas `Smart` mode will engage PQ only upon backpressure from Destinations.

Life Without PQ

On the Destination side, you can configure backpressure behavior to one of **Block**, **Drop Events**, or (on Destinations that support it) **Persistent Queue**. In **Block** mode, the output will refuse to accept new data until the receiver is ready.

The system will back propagate block "signals" all the way back to the sender (assuming that the sender supports backpressure, too). In general, TCP-based senders support backpressure, but this is not a guarantee: Each upstream application's developer is responsible for ensuring that the application stops sending data once Cribl Edge stops sending TCP acknowledgments back to it.

In **Drop** mode, the Destination will discard new events until the receiver is ready. In some environments, the in-memory queues and their block/drop behavior are acceptable.

PQ = Durability

Persistent queues serve environments where more durability is required (e.g., outages last longer than memory queues can sustain), or where upstream senders do not support backpressure (e.g., ephemeral/network senders).

Engaging persistent queues in these scenarios can help minimize data loss. Once the in-memory queue is full, the Cribl Edge Destination will write its data to disk. Then, when the receiver is ready, the output will start draining the queues.

By default, after data flow is re-established, Cribl Edge will forward events in FIFO (first in, first out) order: It will send out earlier queued events before newly arriving events. In Cribl Edge 4.1 and later, you can instead prioritize new events by disabling Destinations' **Strict ordering** control.

Source and/or Destination PQ?

If your Source(s) and Destination(s) both support persistent queues, which side should you enable? If you prioritize maximum data retention and delivery over performance, Cribl recommends that you enable both Source PQ and Destination PQ.

When PQ is engaged, throughput will be somewhat slower. But in exchange for this extra latency, you'll minimize your risk of data loss.

> Because of this latency penalty, it is redundant to enable PQ on a Source whose upstream sender is configured to safeguard events in its own disk buffer.

Persistent Queue Details and Constraints

Persistent queues are:

- Available on Push Sources.

- Available on the output side (i.e., after processing) of all streaming Destinations, with these exceptions: Syslog and Graphite (when you select UDP as the outbound protocol) and SNMP Trap.
- Implemented at the Worker Process level, with independent sizing configuration and dynamic engagement per Worker Process.
- With load-balanced Destinations (Splunk Load Balanced, Splunk HEC, Elasticsearch, TCP JSON, and Syslog with TCP), engaged only when **all of the Destination's receivers** are blocking data flow. (Here, a single live receiver will prevent PQ from engaging on the corresponding Destination.)
- On Destinations, engaged only when receivers are down, unreachable, blocking, or throwing a serious error (such as a connection reset). Destination-side PQ is not designed to engage when receivers' data consumption rate simply slows down.
- Drained when at least one receiver can accept data.
- Capable of orphaning events if there are files on disk when a Source or Destination is turned off or if the number of Worker Processes is reduced.
- Not infinite in size. I.e., if data cannot be delivered out, you might run out of disk space.
- Not able to fully protect in cases of application failure. E.g., in-memory data might get lost if a crash occurs.
- Not able to protect in cases of hardware failure. E.g., disk failure, corruption, or machine/host loss.
- TLS-encrypted only for data in flight, and only on Destinations where TLS is supported and enabled. To encrypt data at rest, including disk writes/reads, you must configure encryption on the underlying storage volume(s).

Configuring Persistent Queueing

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 Cloud Workers

Persistent queuing is supported for Cribl-managed Workers in Cribl.Cloud deployments with an Enterprise plan. Configuration is considerably simpler with Cribl-managed Workers than with hybrid and on-prem Workers (both covered below). You configure PQ individually for each Source and Destination that supports it.

Sources

1. Click a Source to open its configuration modal.
2. Select **Persistent Queue Settings**.
3. Toggle the **Enable Persistent Queue** slider on.

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 no compression is applied to the queue. This limit is not configurable.

> On Cribl-managed Workers, the **Persistent Queue Settings** tab exposes only the **Enable Persistent Queue** toggle. It omits the configuration options available with on-prem or hybrid Workers.

Destinations

1. Click a Destination to open its configuration modal.
2. Set **Backpressure behavior** to `Persistent Queueing`.

When 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. This limit is not configurable.

To disable persistent queuing, select **Persistent Queue Settings**, and then select **Clear Persistent Queue**.

> On Cribl-managed Workers, the **Persistent Queue Settings** tab exposes only the **Clear Persistent Queue** button. It omits the configuration options available with on-prem or hybrid Workers. If the queue fills up, Cribl Edge will block outbound data.

Logs

Cribl internal logs for Cloud-managed Workers might include entries labeled `Revived PQ Worker Process`. These indicate a Cribl-managed Worker newly assigned to drain a Persistent Queue that is left over from a shut-down Worker Node. They can be used to troubleshoot Cloud PQ issues.

Configuring PQ for On-Prem and Hybrid Workers

Persistent queuing is supported for Workers in on-prem deployments with a Standard license, and for hybrid Cribl.Cloud Workers with an Enterprise plan. You configure persistent queueing individually for each Source and Destination that supports it.

On Sources:

1. Click a Source to open its configuration modal.
2. Select **Persistent Queue Settings**.

3. Toggle the **Enable Persistent Queue** slider on.

On Destinations:

1. Click 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 Configure Cribl Stream to Avoid Data Loss With Persistent Queuing 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 Source. Defaults to `1000`.

Commit frequency: The number of events to send downstream before committing that Stream has read them. Defaults to `42`.

Always On versus Smart Mode

As of version 4.1, Source-side PQ's default mode changed from `Smart` to `Always on`, for greater reliability. This 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 find `Smart` mode causing Source-side PQ to frequently engage, and PQ is configured with a large **Max file size** (e.g., 1 GB), Cribl recommends either of two remedies:

- Switch the PQ configuration to `Always On`. This will avoid triggering backpressure and potential data loss on senders. (However, it might require provisioning more disk space for the queue.)

- 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, 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, Cribl Edge will stop queueing data, and will apply the **Queue-full behavior**. Enter a numeral with units of KB, MB, etc. If not specified, the implicit `0` default will enable Cribl Edge to fill all available disk space on the volume.

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.

> ### 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

Queue-full behavior: Determines 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 button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear. (Appears only after **Output ID** has been defined.)

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. 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.

Minimum Free Disk Space

For queuing to operate properly, you must provide sufficient disk space. You configure the minimum disk space in **Settings** > **Global 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 5GB. Be sure to set this on your Edge Nodes (rather than on the Leader Node) when in distributed mode.

Persistent Queue Support by Destination Type

Persistent Queues support, behavior, and triggers vary by Destination type, as summarized below.

HTTP-Based Destinations

HTTP-based Destinations handle backpressure based on HTTP response codes. The following conditions will trigger PQ:

1. Connection failure.
2. HTTP `500` responses.
3. Data overload – sending more data than the Destination will accept.

HTTP `400` response errors will not engage PQ, and Cribl Edge will simply drop corresponding events. Cribl Edge cannot retry these requests, because they have been flagged as "bad," and would just fail again. If you see `400` errors, these often indicate a need to correct your Destination's configuration.

HTTP-based Destinations include:

- WebHook
- New Relic Logs & Metrics
- New Relic Events
- Open Telemetry
- Sumo Logic
- DataDog
- Elastic
- Honeycomb
- Prometheus
- Splunk HEC
- Signal FX
- Wavefront
- Google Chronicle
- Loki

TCP Load-Balanced Destinations

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. The following conditions will trigger PQ:

1. Connection errors on **all** receivers.

   > As long as even one of the Destination's receivers is healthy, Cribl Edge will redirect data to that receiver, and will **not** engage PQ.

2. Data overload – sending more data than the Destination will accept.

TCP load-balanced Destinations include:

- Splunk Load Balanced
- TCP JSON
- Syslog

Destinations Without PQ Support

Filesystem-based Destinations, and Destinations using the UDP protocol, do not support PQ. These include:

- Amazon S3
- Filesystem
- Azure Blob Storage
- Google Cloud Storage
- SNMP Trap
- StastsD (with UDP)
- Statsd Extended (with UDP)
- Syslog (with UDP)

Filesystem-based Destinations do not support PQ because they already persist events to disk, before sending them to their final destinations.

UDP-based Destinations do not support PQ because the protocol is not reliable. Cribl Edge gets no indication whether the receiver received an event.

Other Destinations

Other Destinations with a single receiver generally engage PQ based on these trigger conditions:

1. Connection errors.
2. Fail to send an event (for any reason).

Getting Notified about PQ Status

With an Enterprise or Standard license, you can configure Notifications to be sent when Persistent Queue files exceed a configurable percentage of allocated storage, or when they reach the queue-full state described above.

These Notifications always appear in Cribl Edge's UI and internal logs, and you can also send them to external systems. For setup details, see Destination-State Notifications.

6.7. Fleet Settings

By selecting **Settings** > **Fleet Settings** from Cribl Edge's top nav, you can configure the following settings per Fleet.

General Settings

You can configure the following options in **Fleet Configuration**:

Description: Optionally, add or edit a description of the Fleet's purpose.

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**:

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 over the preceding 1 minute exceed this threshold, Cribl Edge will reject new connections until the CPU load stabilizes. Defaults to `0` % (no throttling). Enter a percentage between `1` – `100` to enable throttling.

> You 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.)

Single-threaded mode (V8): Limits the JavaScript V8 engine to a single thread. Consult Cribl Support before toggling this off.

Other Settings

This page's remaining options work essentially the same way as their **Global Settings** counterparts. Use the following links for details about: logging levels/redactions, access management security, scripts, and diagnostics.

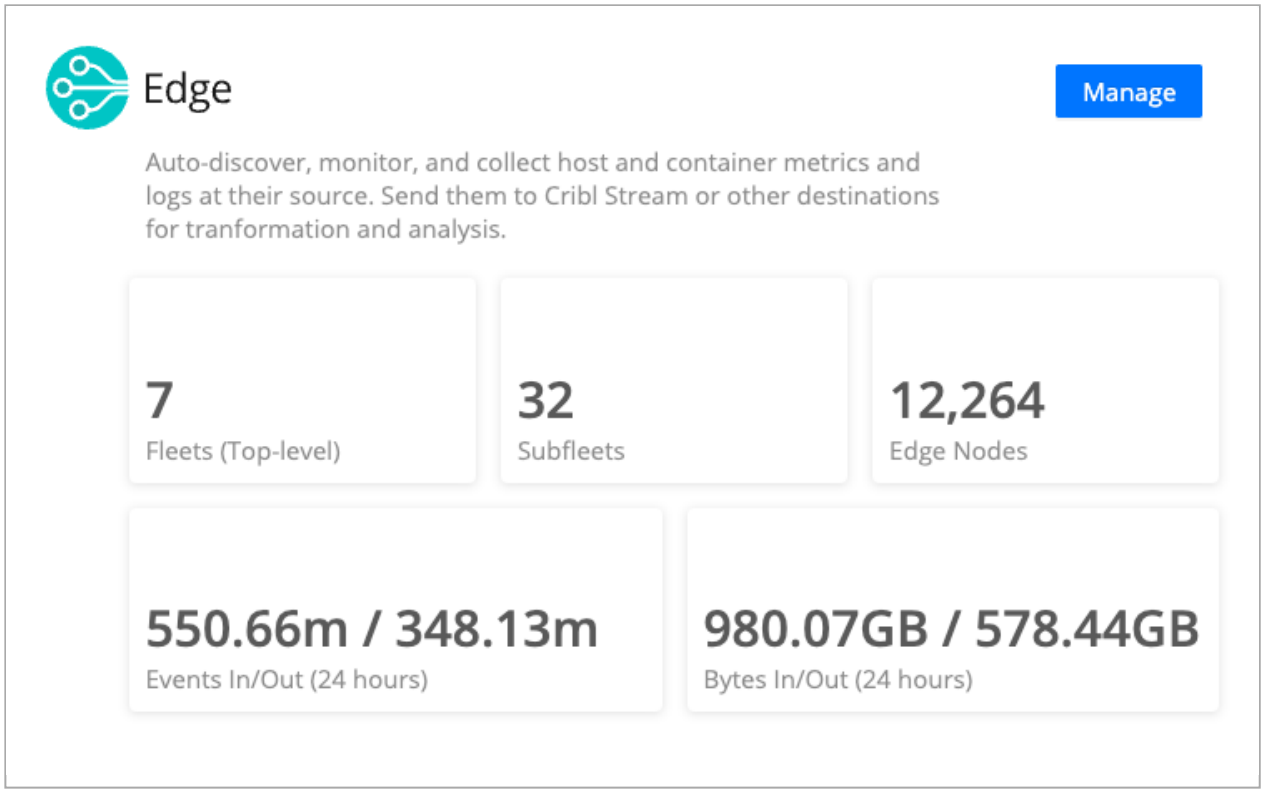
6.8. Monitoring and Notifications

6.8.1. Monitoring Health and Metrics

To get an operational view of your Cribl Edge deployment, consult the following resources.

Landing Pages

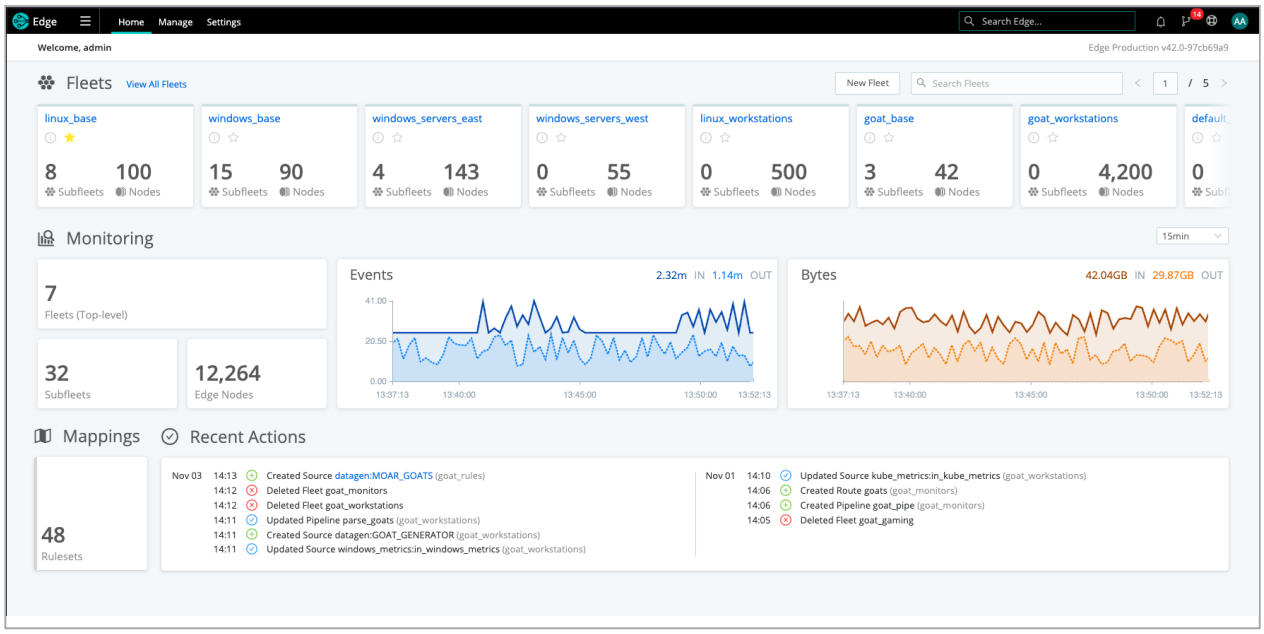
When you first log into your distributed deployment, you will be greeted with tiles prompting you to choose a product. The Cribl Edge tile displays basic configuration details, including number of Fleets, Subfleets, Edge Nodes, and events and bytes over time.



Manage Edge Tile

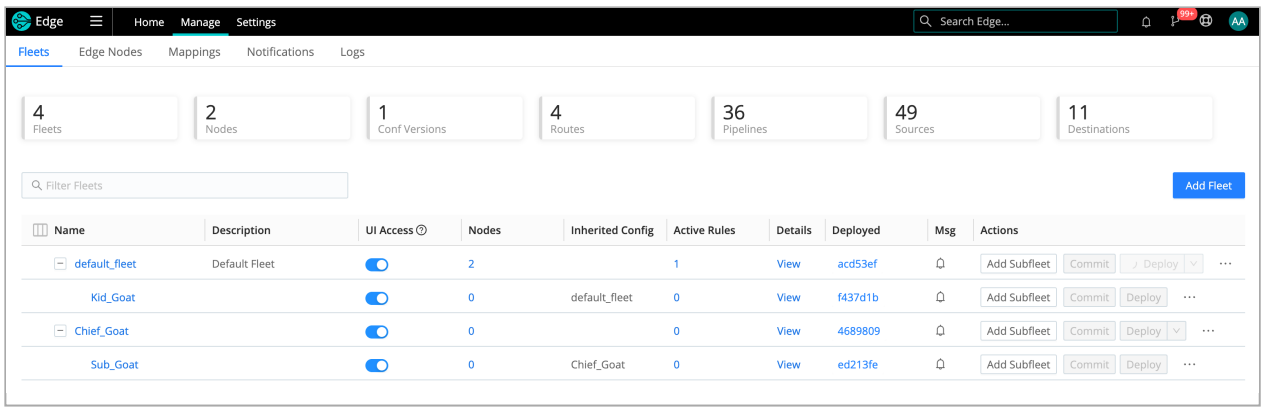
Select **Manage** to navigate to the Cribl Edge **Home** page, which shows aggregate data for all Fleets, Subfleets and Edge Nodes. The charts display information about traffic in and out of the system.

On top of the **Bytes** chart, 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` (covering the preceding 24 hours).



Cribl Edge Home page

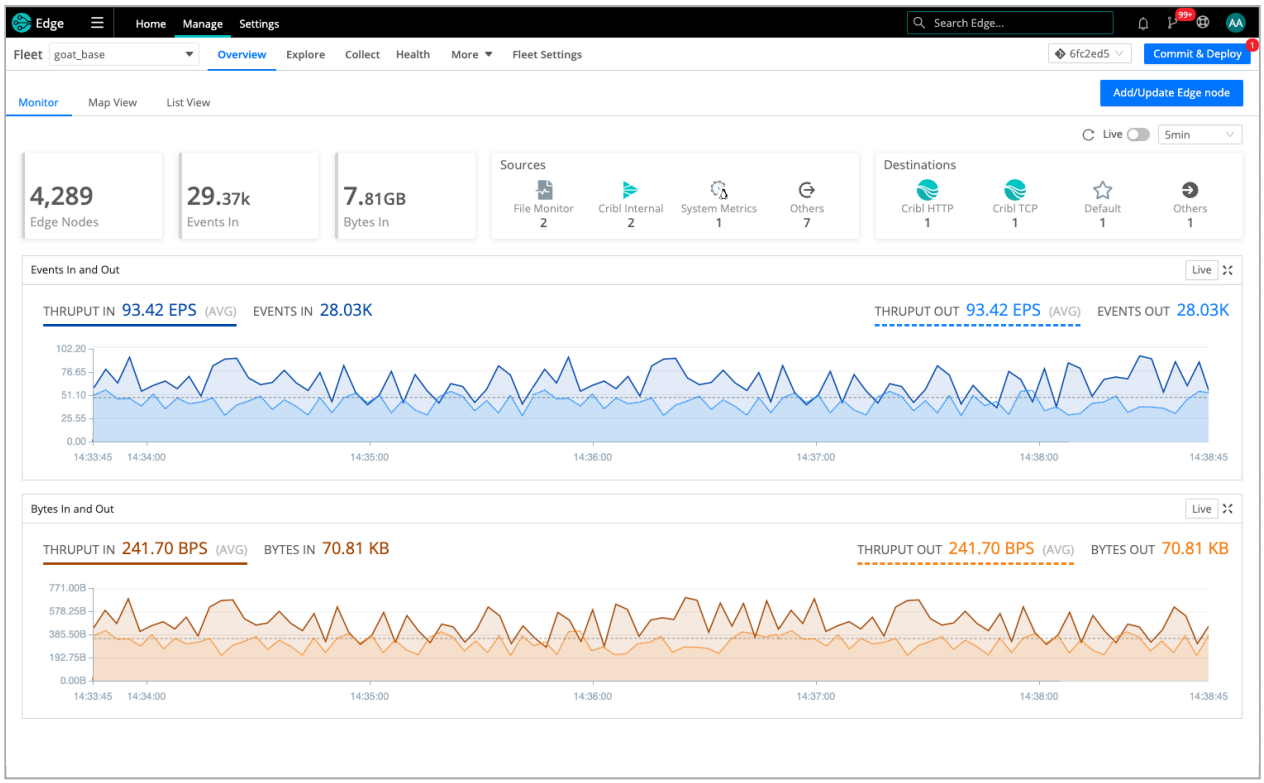
Select **Manage** from the top nav to view the **Fleets Landing** page. Here you can access the tabs for more information about your **Fleets** (and **Subfleets**), **Edge Nodes**, **Mappings**, **Notifications**, and **Logs**.



Manage Fleets

The **Manage Fleets** page gives you access to more information about your **Fleets** (and **Subfleets**), **Edge Nodes**, **Mappings**, **Notifications**, and **Logs**.

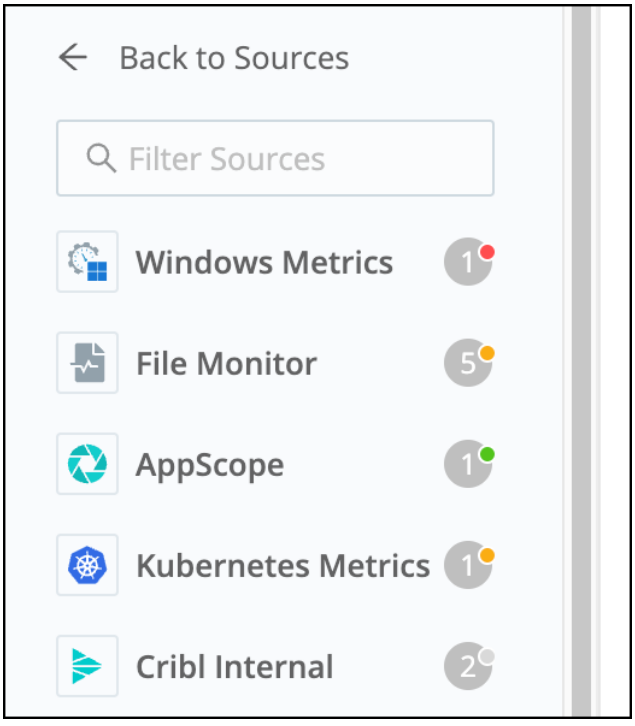
You can click a **Fleet** link to isolate individual Fleets, or use the **Search** bar to locate your Fleet.



Fleet Landing page

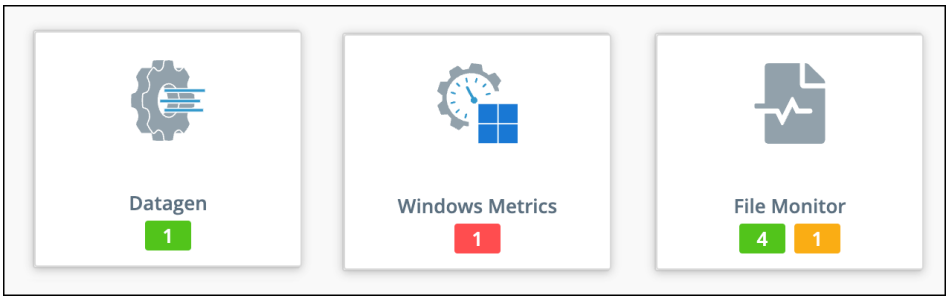
Source and Destination Health Status

When you add or manage configured Sources and Destinations, you'll see a colored health status (if enabled). In the list view, this 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.

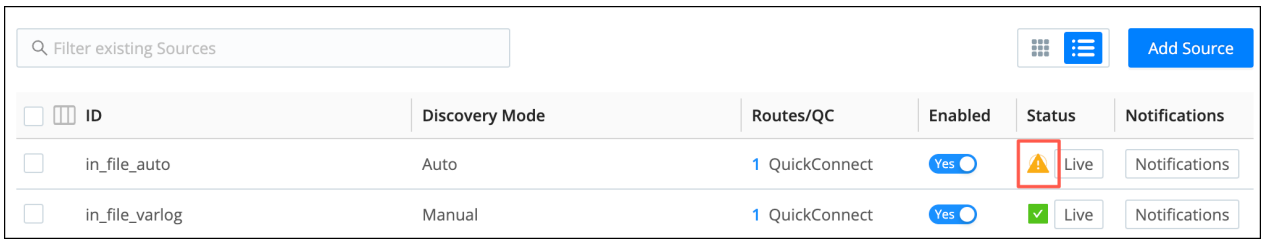


Source tile view

The colors have the following meanings for each Source and Destination:

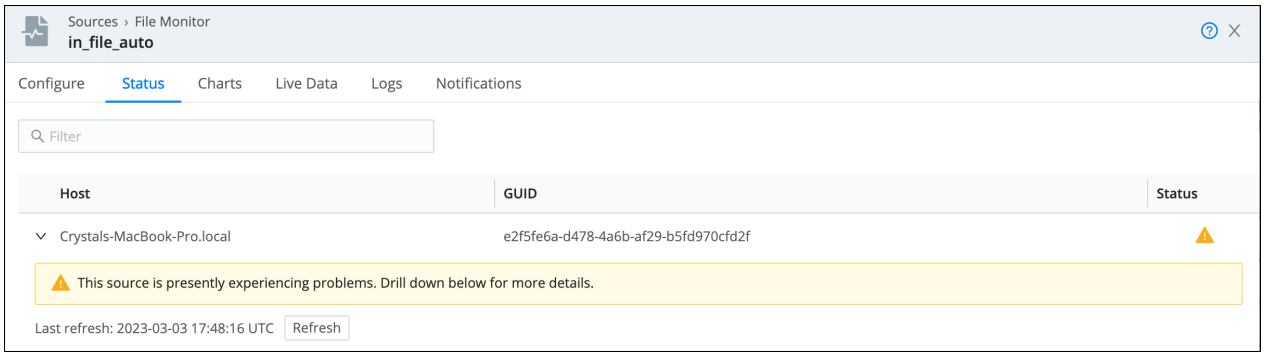
- **Green**: Working properly
- **Orange**: Experiencing problems
- **Red**: Experiencing severe problems
- **Gray**: Not enabled

To learn more about the health status of any Source or Destination, click its name to open the list of existing Sources or Destinations, and then click the Status icon for the one you want to examine.



Health status icon

In the modal that opens, click the Status icon again to see more details.



Health status details

Enabling Health Status

Health status must be enabled independently for each fleet in Fleet Settings. To enable health status for a Fleet:

1. Select the Fleet you want to manage.
2. On the **Manage** page, 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.

> If **Minimal** is selected, health status will not be enabled for the Fleet and health icons will appear white for all Sources and Destinations.

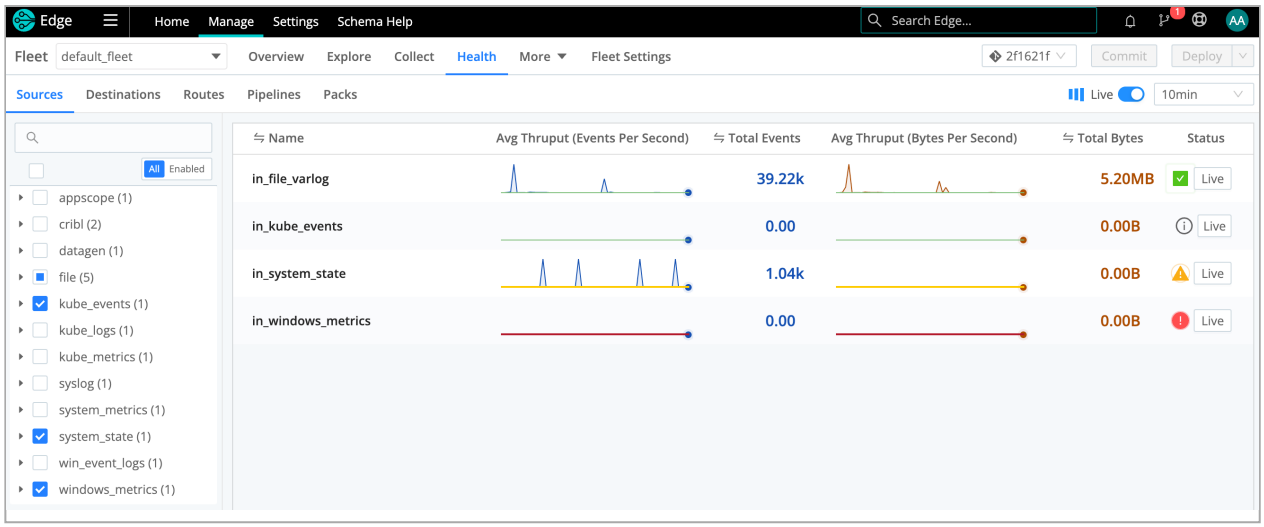
To learn more about these metrics options, see Controlling Metrics.

Health Status

You can isolate throughput for any of the following:

- Sources
- Destinations
- Routes
- Pipelines
- Packs
- Data Fields (Edge Nodes only)

For Fleets, go to the **Health** tab and click the tab you want. For Edge Nodes, go to the Edge Node's **Health** tab, and then select an option on the **Data** submenu.



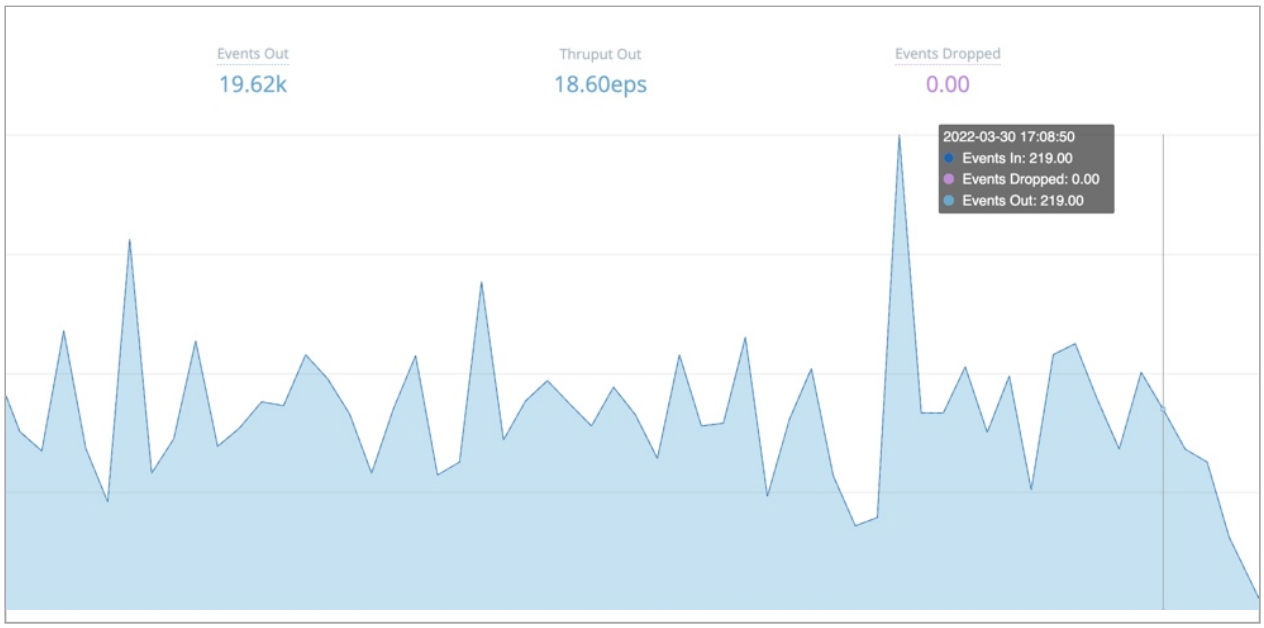
Manage > Health tab

Dense displays are condensed to sparklines for legibility.

The **Status** column displays the health of each resource with a colored icon. Click 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. You can enable health status by going to **Fleet Settings** > **General Settings** > **Metrics** and selecting **Basic** or **All** under **Metrics to send from Edge Nodes**. See Controlling Metrics to learn more about these options.

Hover over the right edge to display Maximize buttons that you can click to zoom these up to detailed graphs.

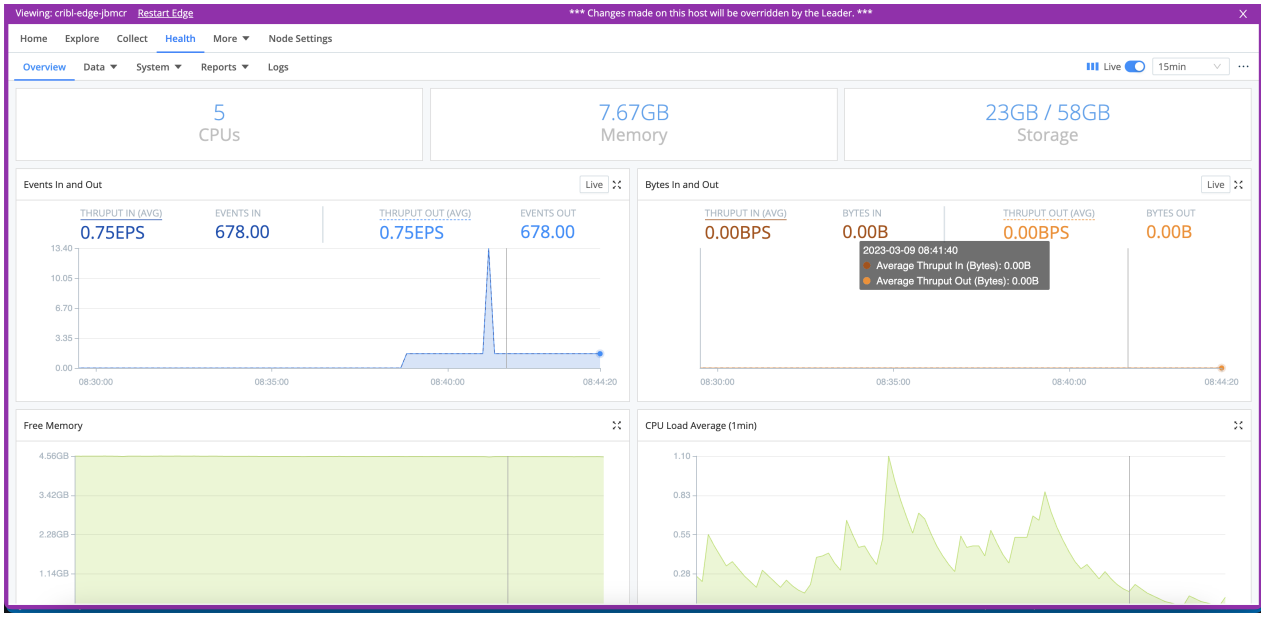
You can hover over an expanded graph fly-out to display further details.



Throughput details

Edge Node Health

To review the health status of an Edge Node, teleport into the Node. From the **Fleet** landing page, click the **List View** tab, and then click the **Edge Node GUID** link to teleport from the Leader into the Edge Node. From the top nav of the Edge Node, click the **Health** tab for monitoring details.



Health Overview

> Metrics are displayed in the Cribl Edge UI in units of KB, MB, GB, and TB. At each level, these are multiples of `1024`.

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.

System Monitoring

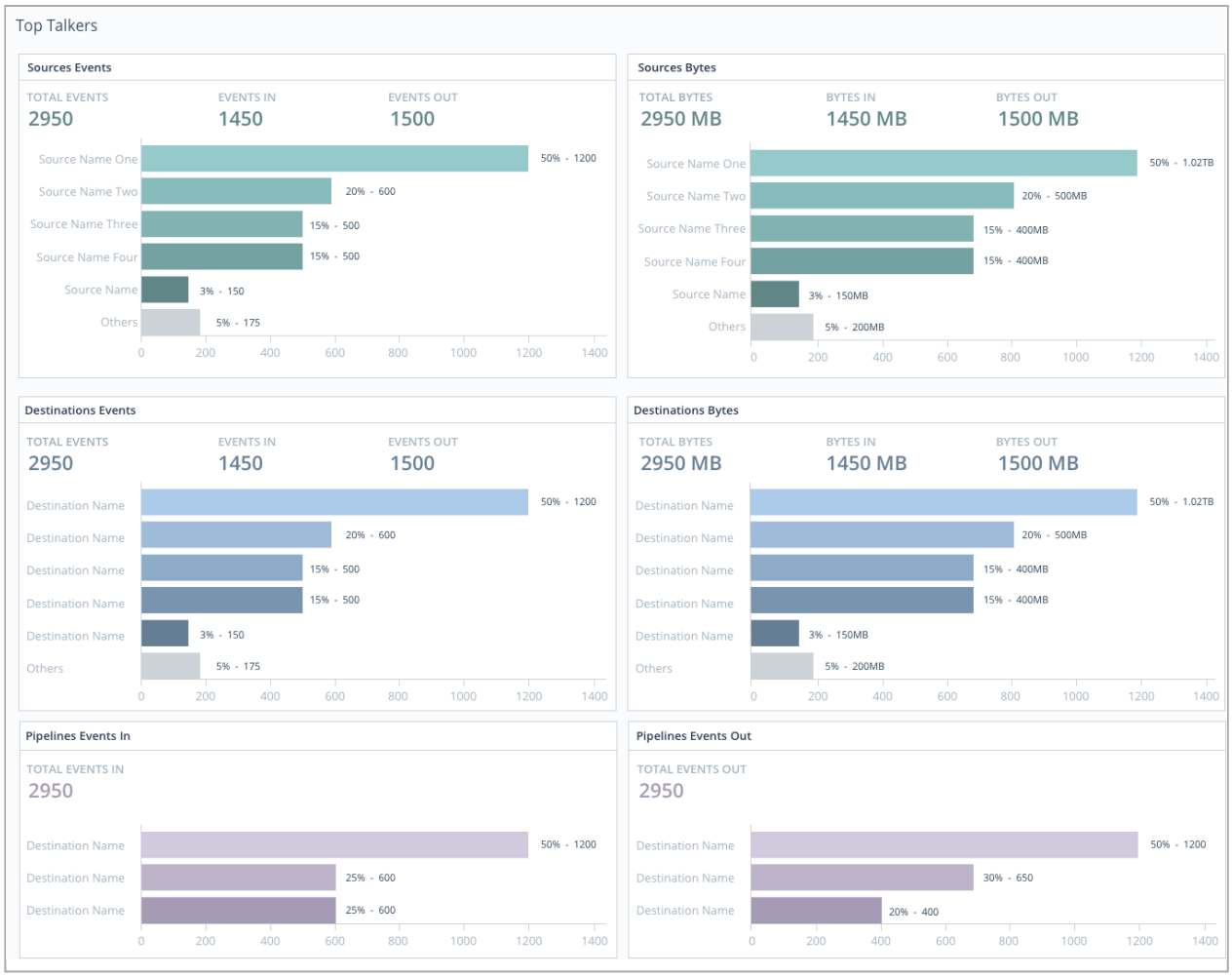
From the **Health** page's top nav, open the **System** submenu to isolate throughput for:

- Queues (Sources)
- Queues (Destinations)

For details, see Persistent Queues.

Reports/Top Talkers

Select **Health** > **Reports / Top Talkers** > **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

Select **Logs** from the **Health** page's top nav. Cribl Edge's internal logs and internal metrics provide comprehensive information about an instance's status/health, inputs, outputs, Pipelines, Routes, Functions, and traffic.

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.

Cribl Edge rotates logs every 5 MB, keeping the most recent 5 logs. 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.

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.

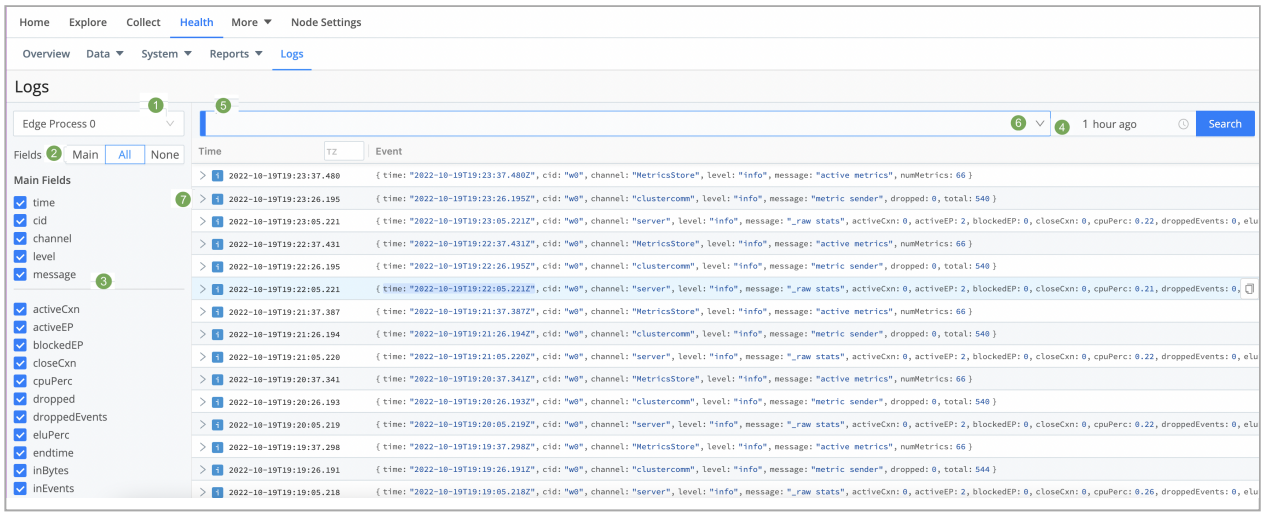
For recommendations about useful Cribl metrics to monitor, see Internal Metrics.

> **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 (controls highlighted)

1. **Log file selector**: Choose the types of logs to view.

2. **Fields selector**: Click the **Main | All | 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 are displayed 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. Click the Search box's history arrow (right side) to retrieve recent queries.

7. The Results pane displays most-recent events first. Each event's icon is color-coded to match the event's severity level.

   Click 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.

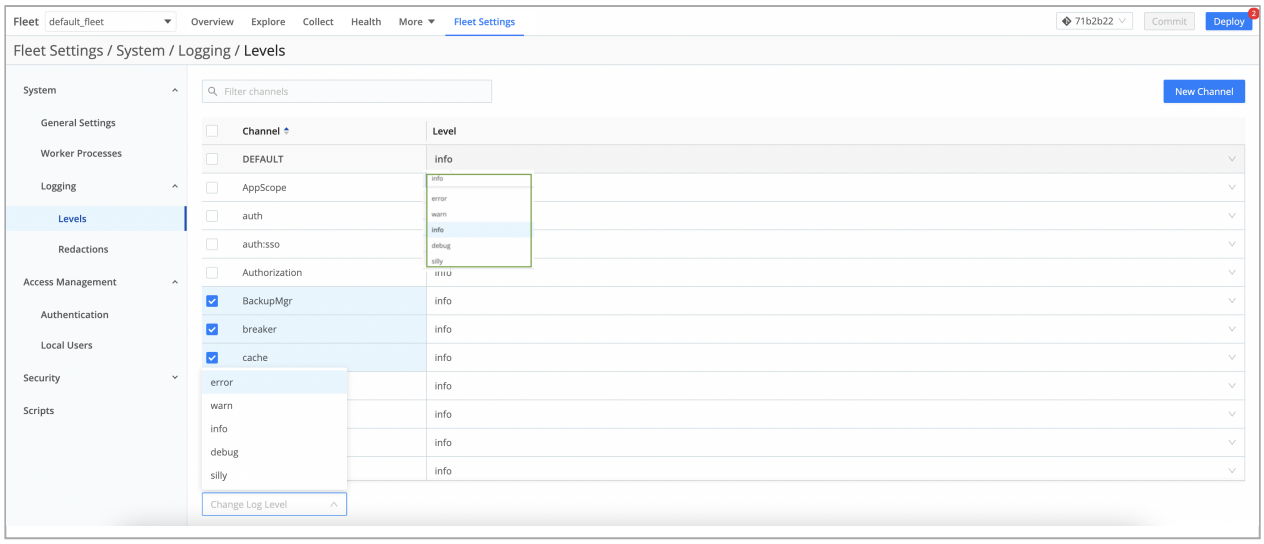
Change Logging Levels

To adjust logging levels:

- From Cribl Edge's top nav, select **Settings** > **System > Logging > Levels**.
- To configure log levels on a Fleet-level, click **Manage**, then select the Fleet you want to configure. Next, select **Fleet Settings** > **System > Logging > Levels**.
- To adjust settings on a a specific Edge Node, teleport into the Node, then select **Node Settings** > **System > Logging > Levels**.

On the resulting **Manage Logging Levels** page, you can:

- Modify one channel by clicking 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 clicking the **Change log level** drop-down at the bottom of the page. (Bottom of composite screenshot below.) You can select all channels at once by clicking the top check box. You can search for channels at top right.



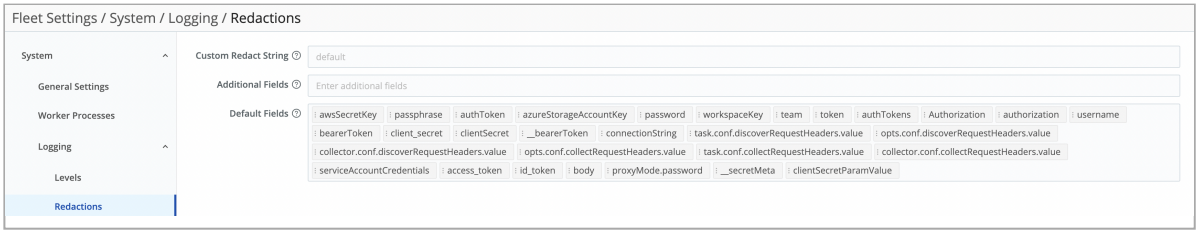
Manage Logging Levels page

> The `silly` (ultra-verbose) logging level is available for all channels. However, it provides additional metrics information only for inputs.

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** (top nav) > **System > Logging > Redactions**.
- In a distributed deployment, first click **Manage**, then select the Fleet you want to configure. Next, select **Fleet Settings** > **System > Logging > Redactions**.



Redact Internal Log Fields page

It's easiest to understand the resulting **Redact Internal Log Fields** page's fields from bottom to top:

- **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.

HEALTH CHECK ENDPOINT	HEALTHY RESPONSE	CRIBL EDGE VERSION
`curl http(s)://<host>:<port>/api/v1/health`	`{"status":"healthy"}`	Through 2.4.3
`curl http(s)://<host>:<port>/api/v1/health`	`{"status":"healthy","startTime":1617814717110}` (see details below)	2.4.4 and later

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 examples, `<port>` stands for the API port (by default, `9420`).

6.8.1.1. 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 a Worker/Edge Node is having a problem, a Node is down, a Destination is down, a Source stops providing incoming data, etc.

Cribl Edge reports its internal metrics within the Cribl Edge UI, in the same way that it reports internal logs at **Monitoring** > **Logs** (Cribl Stream) and **Health** > **Logs** (Cribl Edge). To expose metrics for capture or routing, enable the **Cribl Internal** Source > **CriblMetrics** section.

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. E.g., 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 }
  ],
```

Worker/Edge Node Resource Metrics

As of Cribl Stream (LogStream) 2.4.4, the Cribl Internal Source reports two useful metrics on individual Worker/Edge Node Processes' resource usage:

- `system.cpu_perc` – CPU percentage usage.
- `system.mem_rss` – RAM usage.

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 Destination, using 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 Destination, per Worker/Edge NodeProcess.
- `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.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.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.

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.

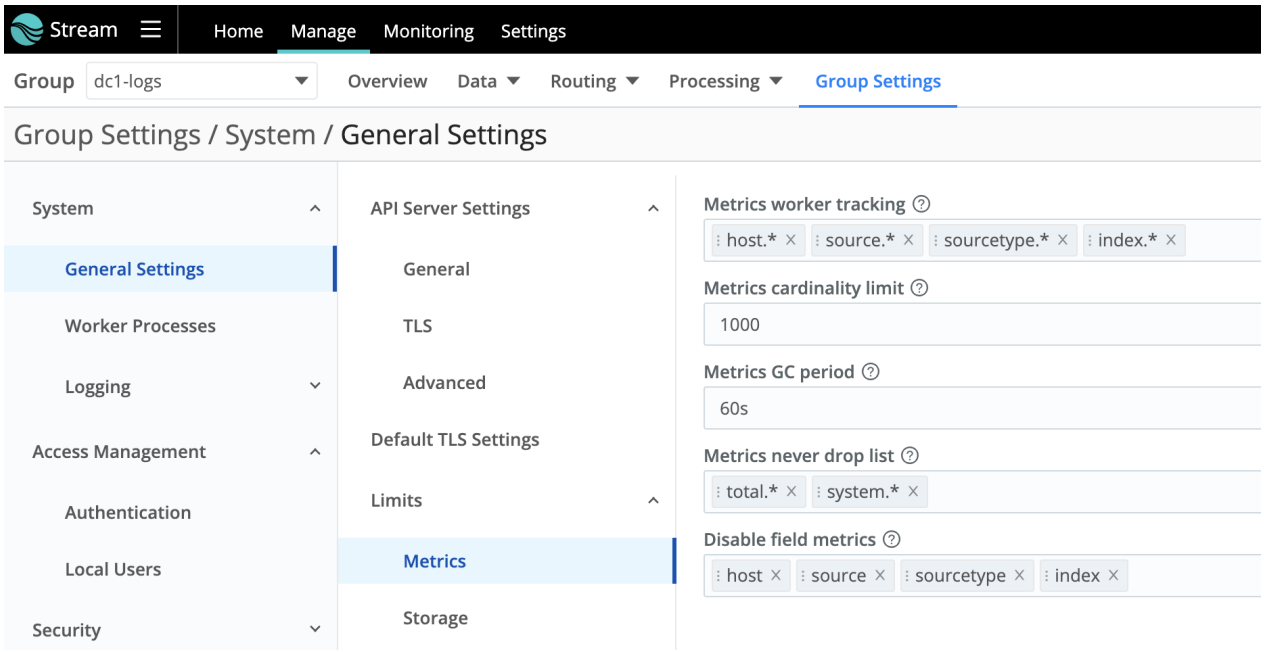
Controlling Metrics

You can control the volume of internal metrics that Cribl Edge emits by adjusting Metrics Limits, which you access as follows:

- Distributed deployments: [Select a Fleet] > **Group/Fleet Settings** > **System** > **General Settings** > **Limits** > **Metrics**.
- Single-instance deployments: **Settings** > **Global Settings** > **System** > **General Settings** > **Limits** > **Metrics**.

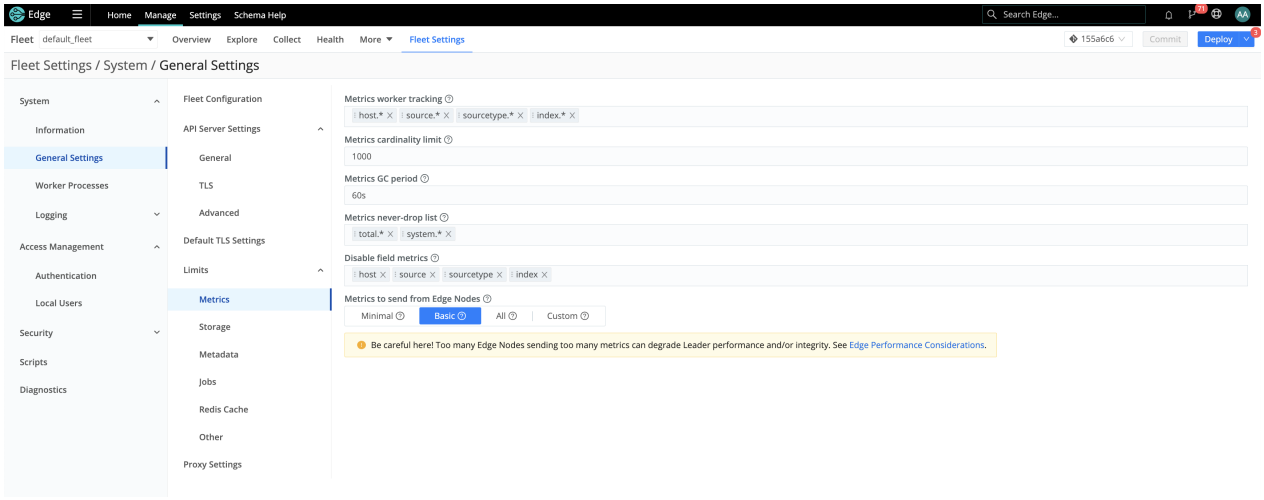
Here, you can adjust metrics' cardinality, metrics to always include (**Metrics never-drop list**), metrics to never include (**Disable field metrics**), and other options.

Here's how these controls look in Cribl Stream:



Metrics settings for Cribl Stream Worker Groups

They're somewhat different in Cribl Edge – the next section for an explanation:



Metrics settings for Cribl Edge Fleets

Controlling 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	Send 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 from Sources or Destinations directly.

Basic (default)	Send metrics required to display all monitoring data available in the Edge UI. Contains all the metrics in the Minimal set, and more.	Sent as aggregate totals per Source/Destination:
		
		`total.in_events`, `total.out_events`, `total.in_bytes`, `total.out_bytes`.
		
		Also, Sent as aggregate totals per Source:
		
		`health.inputs`
		
		Sent as aggregate totals per Destination:
		
		`health.outputs`
		
		Sent as aggregate totals per Route:
		
		`route.in_events`, `route.in_bytes`, `route.out_events`, `route.out_bytes`, `route.dropped_events`
		
		Sent as aggregate totals per Pipeline:
		
		`pipe.in_events`, `pipe.out_events`, `pipe.dropped_events`
		
		Sent as aggregate totals per Pack:
		
		`pack.in_events`, `pack.out_events`, `pack.err_events`, `pack.dropped_events`

All	Send all metrics with no filtering.	All metrics without filtering, including **Basic**, plus Worker/Edge Node Resource Metrics, Persistent Queue Metrics, and Other Internal Metrics.
Custom	Send a customized set of metrics.	All metrics that match a JavaScript expression that you define, plus the metrics in the **Minimal** set.

Every Edge Fleet always sends the **Minimal** metrics; selecting one of the other options only adds more metrics.

Choosing the **Minimal** metrics set has the following effects in the Edge UI:

- Only one UI page, **Manage** > **Overview** > **Monitor**, will be fully populated. For the metrics that the **Minimal** set excludes, other pages will show zero values. This applies to the **Manage** > **Health** pages for **Sources**, **Destinations**, **Routes**, **Pipelines**, and **Packs**.
- In the **Manage** > **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`.

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.

6.8.1.2. Internal Logs

Distributed deployments emit a larger set of logs than single-instance deployments. We'll describe the full set first.

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**

SERVICE NAME	DESCRIPTION	EQUIVALENT ON LOGS PAGE
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**

Worker/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 log in `$CRIBL_HOME/log/worker/N/`, where `N` is the Worker/Edge Node Process ID.

LOGFILE NAME	DESCRIPTION	EQUIVALENT ON LOGS PAGE
`cribl.log`	Messages about the Worker/Edge Node processing data.	**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**

Single-Instance Logs

The `API/main` process emits the same logs that 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 log in `$CRIBL_HOME/log/worker/N/`, where `N` is the Worker/Edge Node Process ID.

LOGFILE NAME	DESCRIPTION	EQUIVALENT ON LOGS PAGE
`cribl.log`	Messages about the Worker/Edge Node processing data.	**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","message":"_raw
stats","inEvents":307965,"outEvents":495848,"inBytes":52756162,"outBytes":83028013,"star
ttime":1668703980,"endtime":1668704040,"activeCxn":0,"openCxn":0,"closeCxn":0,"rejectCxn
":0,"abortCxn":0,"pqInEvents":62000,"pqOutEvents":114591,"pqInBytes":12163896,"pqOutByte
s":22481509,"pqTotalBytes":480467058,"droppedEvents":0,"tasksStarted":6,"tasksCompleted"
:6,"activeEP":9,"blockedEP":0,"cpuPerc":101.09,"eluPerc":97.81,"mem":
{"heap":277,"ext":46,"rss":453}}
```


Field Descriptions

FIELD	DESCRIPTION
`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.
`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. (E.g., when using the Aggregation Function.)
`inBytes`	Number of bytes received from all Sources (based only off `_raw`).
`outBytes`	Number of bytes sent to all Destinations (based only off `_raw`).
`startTime`	The beginning of the timespan represented by these metrics.
`endTime`	The end of the timespan represented by these metrics. (Will always be 60 seconds after `startTime`.)
`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.)
`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`.
`closeCxn`	Number of TCP connections that were closed.
`pqInEvents`	Number of events that were written to Persistent Queues, across all Destinations.
`pqOutEvents`	Number of events that were flushed from Persistent Queues, across all Destinations.
`pqInBytes`	Number of bytes that were written to Persistent Queues, across all Destinations.
`pqOutBytes`	Number of bytes that were flushed from Persistent Queues, across all Destinations.
`pqTotalBytes`	Amount of data currently stored in Persistent Queues, across all Destinations.
`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.
`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).
`cpuPerc`	CPU utilization from the combined user and system activity over the last 60 seconds.

FIELD	DESCRIPTION
`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.
`mem.heap`	Heap section of process memory, in MB.
`mem.ext`	External section of process memory, in MB.
`mem.rss`	Resident set size section of process memory, in MB.
`tasksStarted`	The number of tasks the process started for all collection jobs for which it was executing tasks.
`tasksCompleted`	The number of tasks the process has started and completed for all collection jobs for which it was executing tasks.

6.8.2. Notifications

In Cribl Stream (LogStream) 3.1 or later, and all Cribl Edge versions, you can configure Notifications about:

- Sources and Collectors that report abnormally high or low data flow rates.
- Sources and Collectors that report no data flow.
- Destinations that report errors.
- Destinations experiencing backpressure.
- Pending expiration of a Cribl Edge license.

Notifications are not designed to take the place of alerts on your overall infrastructure's health – but they warn you about conditions that could impede expected data flow into and out of Cribl Edge.

> Notifications require an Enterprise or Standard license, without which the configuration options described below will be hidden or disabled in Cribl Edge's UI.

Notifications and Targets

Every Notification is sent to one or more **targets**. By default, any Notification that you configure will have a target of `System Messages`. This means that when a Notification is triggered, it will add an indicator on the top nav's **Messages** button. Click this button to view details in the **Messages** drawer.

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

You can also send any Notification to additional targets, using Cribl Edge's native PagerDuty integration and/or by specifying custom webhooks. For details, see Configuring Targets.

Notifications and RBAC

Notifications work with Cribl Edge's 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, and can create and manage Notifications and targets

Configuring Notifications

Destination-state, Source-/Collector-state, and license-expiration Notifications are configured separately.

Destination-State Notifications

On individual Destinations, you can configure Notifications that will trigger under these conditions:

- Destination Backpressure Activated
- Persistent Queue Usage
- Unhealthy Destination

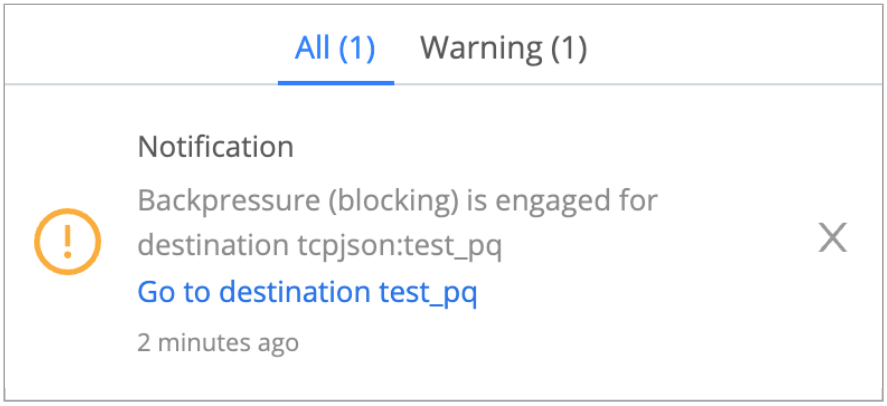
Read on for details about the above conditions, and about how to specify them in a Destination's **Condition** drop-down.

Destination Backpressure Activated

This will generate a Notification reading `Backpressure (<blocking|dropping>) is engaged for Destination <name>` when one of the following events occur:

- 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 outgoing events to block or drop.

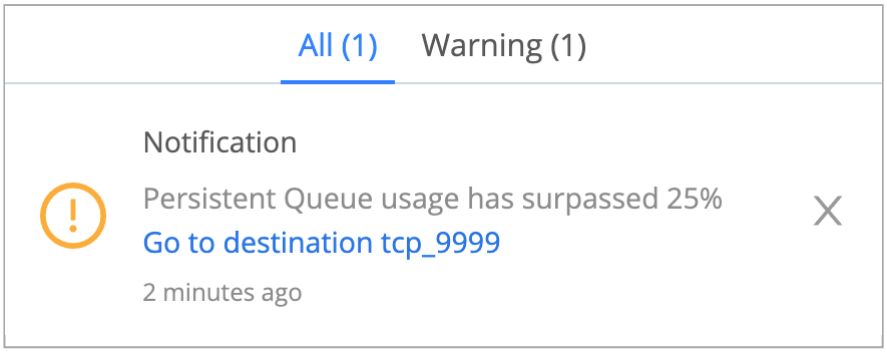
The threshold for the Notification to trigger is: Cribl Edge detected a blocked or dropped state during ≥ 5% of the trailing **Time window** that you configure in the Notification Settings below.



Backpressure notification

Persistent Queue Usage

This will generate a `Persistent Queue usage has surpassed <threshold>%` Notification when PQ files accumulate past the `<threshold>` percentage of capacity that you set in the **Usage threshold** field.



Persistent Queue notification

Unhealthy Destination

This will generate a `Destination <name> is unhealthy` Notification when the Destination's health has been in "red" status (as indicated on the UI's Monitoring page) over the trailing **Time window** that you configure in the Notification Settings below.

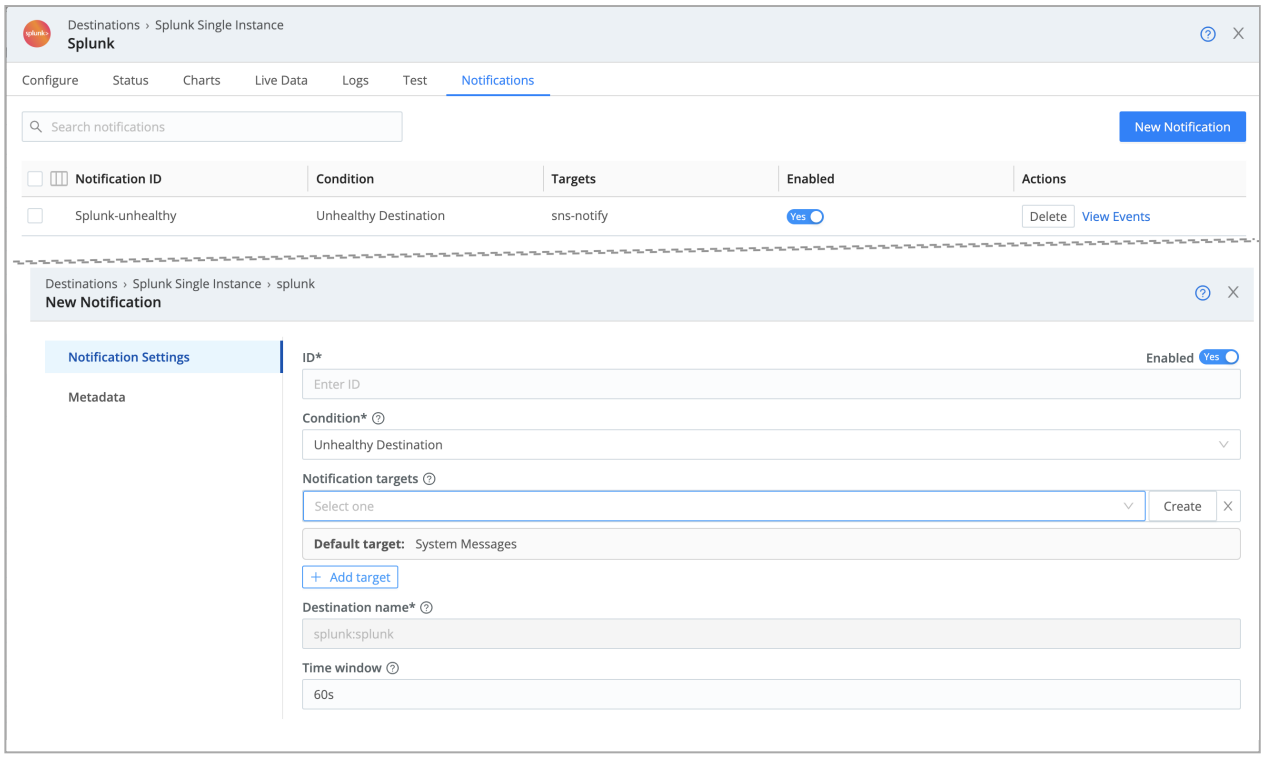
The algorithm has slight variations among Destination types, but red status generally means that ≥ 5% of health checks, aggregated over the **Time window**, reported either:

- An error inhibiting the Destination's normal operation, such as a connection error; or
- For multiple-output Destinations like Splunk Load Balanced or Output Router, > 50% of the Destination senders in an error state.

Configuring Destination Notifications

To start configuring a Destination-state Notification:

1. Configure and save the Destination.
2. Access this Destination's **Notifications** tab. Either:
    - Click the **Notifications** button on the **Manage...Destinations** page's appropriate row, or
    - Reopen the Destination's config modal, and click its **Notifications** tab.
3. Click **Add Notification** to access the **New Notification** modal shown below.



Configuring a Destination Notification (composite screenshot)

The **New Notification** modal provides Notification Settings and Metadata tabs, whose controls are listed in the respective sections below.

Notification Settings

ID: Enter a unique ID for this Notification. (Cribl recommends using a string that will make the Notification's purpose clear.)

Condition: Select either `Unhealthy Destination` or `Destination Backpressure Activated`. (You can set triggers for both conditions on the same Destination, but you must configure them as separate Notifications.)

Notification targets: The **Default target** is always locked to `System Messages`. Click **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.
- Click **Create** to configure a new target. (See Configuring Targets for details.)

Destination name: This is locked to the Destination on which you're setting this Notification.

Time window: Sets the threshold period before the Notification will trigger. E.g., the default `60s` will generate a Notification when the Destination has reported the trigger condition over the past 60 seconds.

(For % trigger conditions, see Destination Backpressure Activated and Unhealthy Destination.) To enter alternative numeric values, append units of `s` for seconds, `m` for minutes, `h` for hours, etc.

Usage threshold: This percentage setting is displayed only when you've set the **Condition** to Persistent Queue Usage.

Metadata

Click **Add field** here to add custom metadata fields to your Notifications, as 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 Notification, you can see Notification events specific to this Destination on the Destination config modal's **Events** tab. (When you set Source-state Notifcations, a corresponding **Events** tab is available on Sources' and Collectors' config modals.) For a comprehensive view of all Notification events, see the systemwide Events Tab.

Recurrence/Expiration

If a Destination-State Notification's trigger condition persists beyond your configured Time window, expect a new notification to be sent once per **Time window** interval.

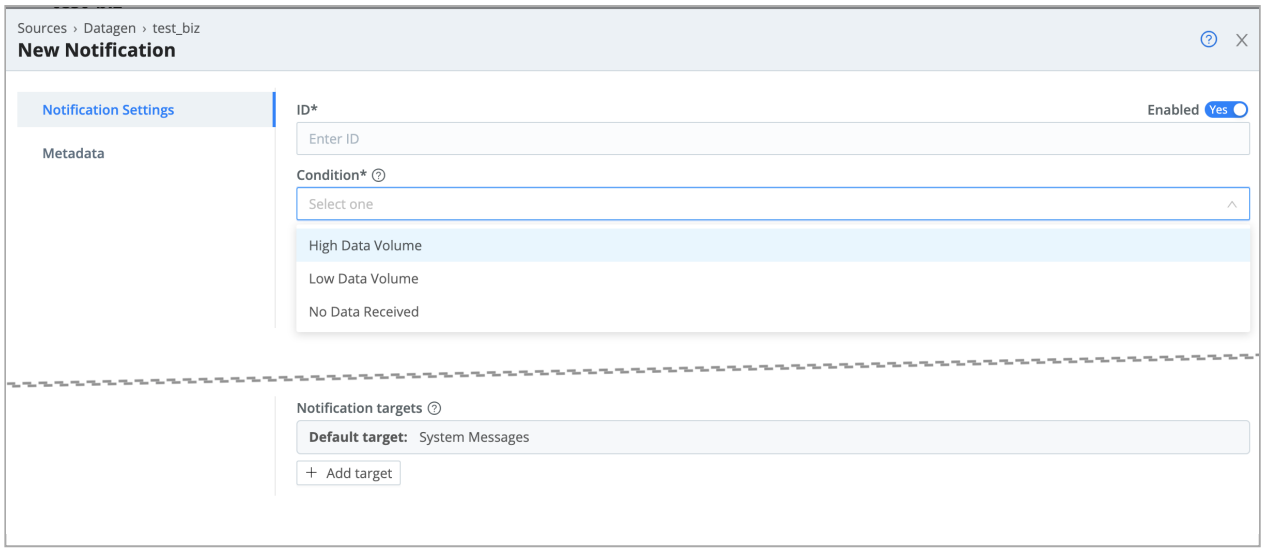
Notifications will cease when the triggering condition clears. There is no explicit "problem cleared" Notification.

Source-State Notifications

In Cribl Edge 3.5 and above, you can configure Notifications on Sources and Collectors to trigger under these conditions:

- High Data Volume
- Low Data Volume
- No Data

You configure these similarly to Destination Notifications: Select the Notification type from the **Condition** drop-down. Each type exposes additional fields, as outlined below.



Configuring a High Data Volume Notification (composite screenshot)

High Data Volume

Select the `High Data Volume` Condition to trigger Notifications when incoming data over your configured **Time window** exceeds your configured **Data volume** threshold. This selection exposes the following fields:

Notification targets: As with Destination-State Notifications, the **Default target** is always locked to `System Messages`. Click **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.
- Click **Create** to configure a new target. (See Configuring Targets for details.)

Source name: This is locked to the Source on which you're setting this Notification.

Time window: As with the corresponding field on Destination-State Notifications, 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, etc.

Data volume: Enter the threshold above which a Notification will trigger. Accepts numerals with units like KB, MB, etc. (e.g.: 4GB).

Low Data Volume

Select the `Low Data Volume` Condition 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` Condition to trigger Notifications when the Source or Collector ingests zero data over your configured **Time window**.

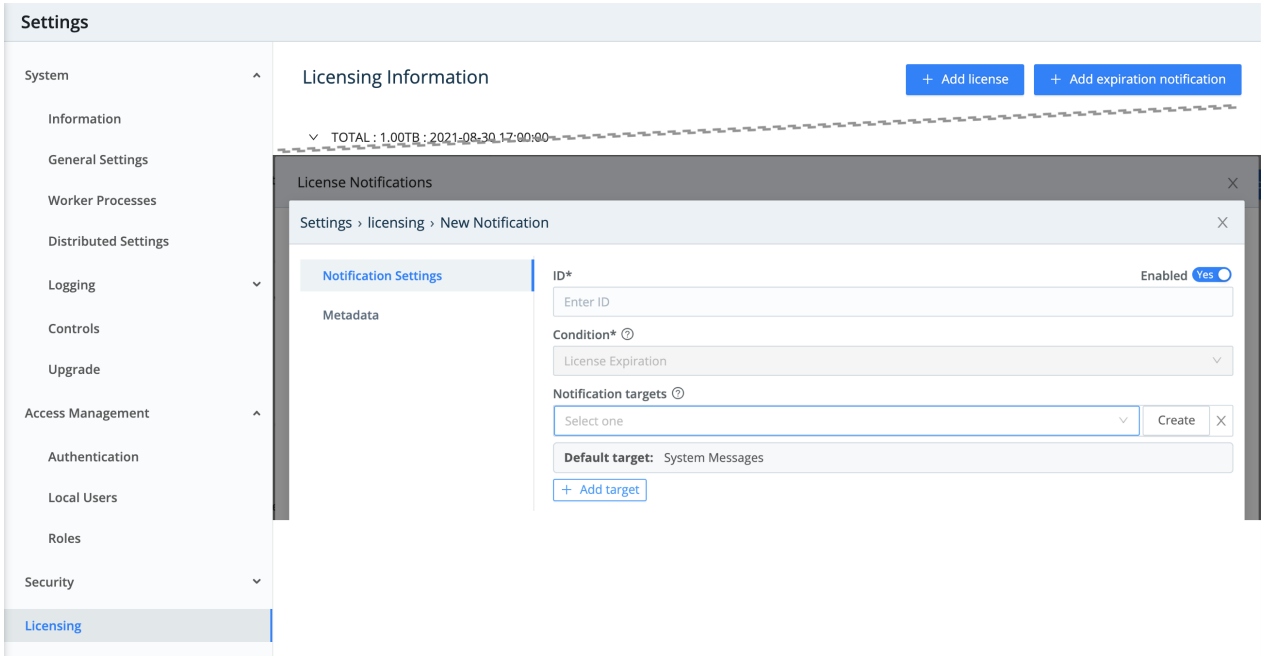
This selection exposes the same additional fields as High Data Volume, except (for obvious reasons) it omits the **Data volume** field – there is no threshold, because this is a binary condition.

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.)

Configuring License-Expiration Notifications

1. From the top nav, select **Manage** > **Settings** > **Licensing**.
2. Click **Add expiration notification** to access the **New Notification** modal shown below.



Configuring an expiration Notification (composite screenshot)

This **New Notification** modal provides Notification Settings and Metadata tabs, with a subset of the controls available in the Destination-unhealthy modal:

Notification Settings

ID: Enter a unique ID for this Notification.

Condition: This modal's triggering condition is locked to `License Expiration`.

Notification targets: The **Default target** is always locked to `System Messages`. Click **Add target** for each additional target that you want to send this Notification to.

- Use the resulting **Notification targets** drop-down to select any target you've already configured.
- Click **Create** to configure a new target. See Configuring Targets for details.

Metadata

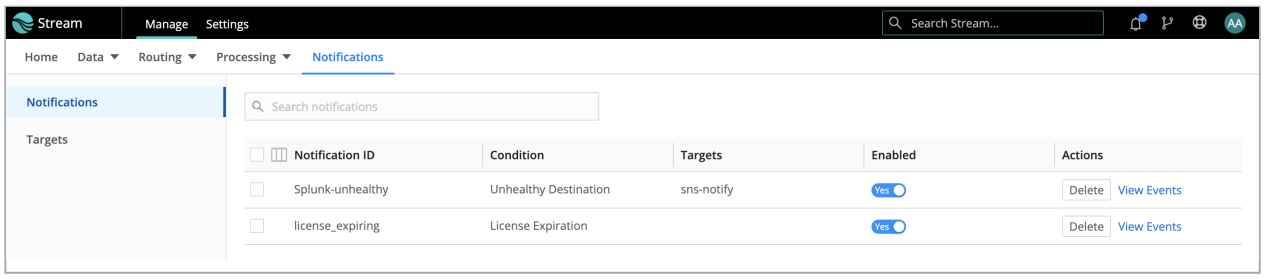
The options here are identical to those on the Destination-unhealthy modal's **Metadata** tab.

Managing Notifications

The **Notifications** page provides global display and controls for all your configured Notifications, targets, and triggered Events – across all Sources, Collectors, Destinations, and all Worker Groups. To access this page: From the top nav, select **Manage** > **Notifications**.

Notifications Tab

This tab lists all your configured Source-state and Destination-unhealthy 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; you can also click on any Notification's row to open and modify its configuration.



Notifications tab

Targets Tab

This tab is where you centrally configure and manage targets that are available across Cribl Edge – to all Source-, Destination-, and license-based Notifications. See Configuring Targets for details.

Events Tab

This tab displays logged events that have been fired by all your configured Notifications. You can filter by search string, and by lookback time.

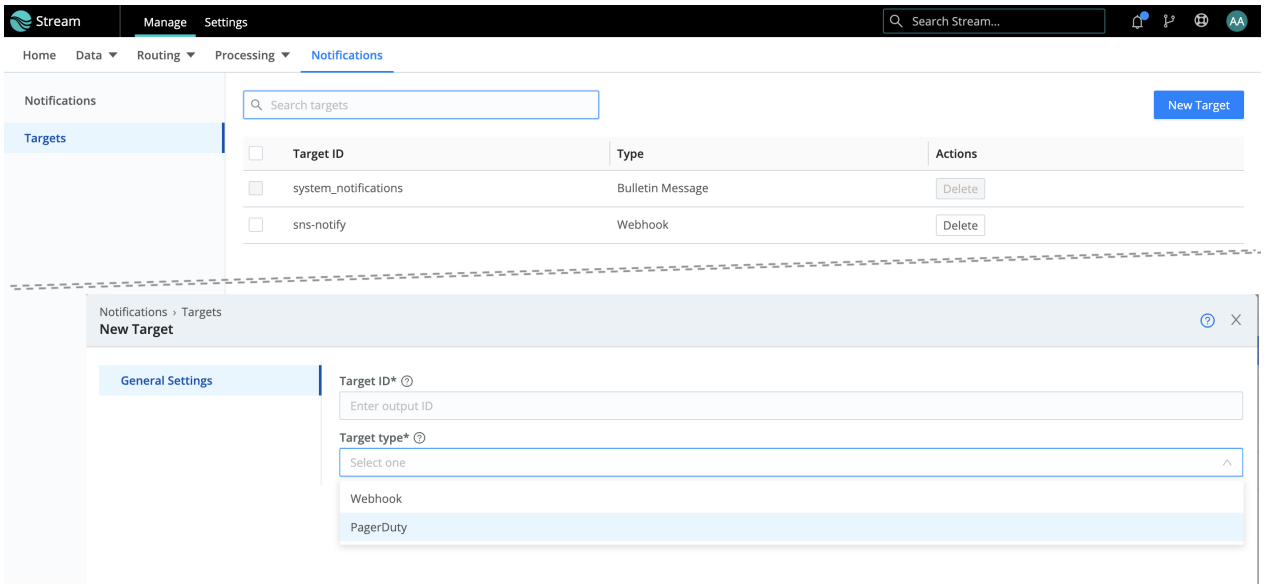
What's Next

In future Cribl Edge releases, Cribl plans to expand the Notifications feature with options to configure additional triggering conditions and time resolutions.

6.8.2.1. Configuring Targets

To add a new Notification target from the Manage Notifications page's **Targets** tab:

1. Click **Add Target** to open the **New Target** modal shown below.
2. Give this target a unique **Target ID**.
3. Set the **Target type** to either PagerDuty or Webhook. Then configure the target according to the corresponding section below.



Adding a new target (composite screenshot)

> Notifications require an Enterprise or Standard license, without which all the target configuration options described on this page will be hidden or disabled in Cribl Edge's UI.

PagerDuty 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 **Target type**: `PagerDuty` to expose the following additional options on the modal's (single) **General Settings** left tab:

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.)

Webhook Targets

With this option, you can send Cribl Edge Notifications to an arbitrary webhook. Select **Target type**: `Webhook` to expose multiple left tabs, with the following configuration options:

General Settings

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. E.g.: `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`.
  - **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.

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.

Processing Settings

The options on this left tab are identical to those on the Webhook Destination's **Processing Settings** tab, with two exceptions:

- The default **System fields** entry here is `cribl_host`.
- You cannot specify a post-processing Pipeline here.

Advanced Settings

The options on this left tab are identical to those on the Webhook Destination's **Advanced Settings** tab.

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.
- `endtime`: End of the time bucket where this condition was reported. All Notifications have this.
- `_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 Backpresssure 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

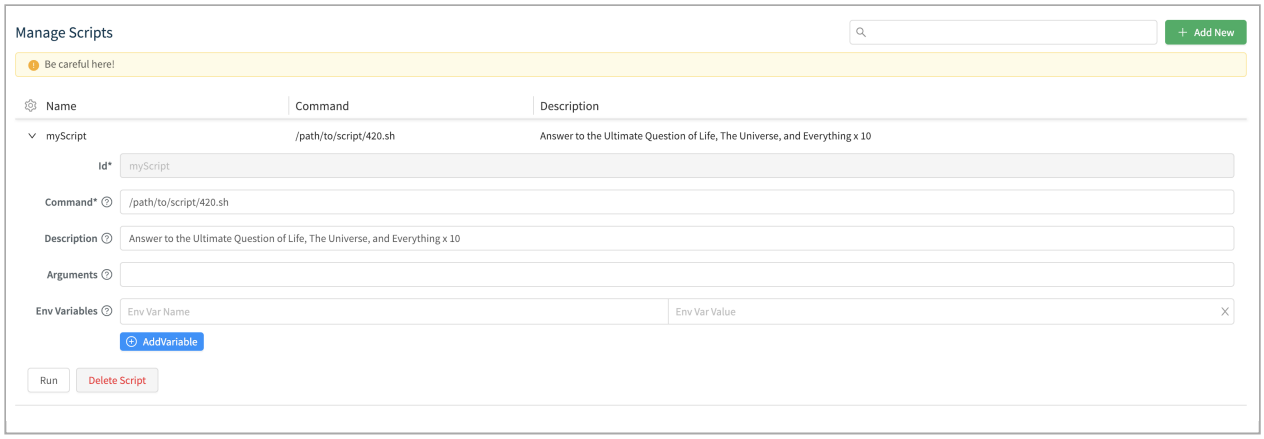
- `severity`: One of "warn" or "fatal".
- `title`: One of: "License expiring soon, data will stop flowing." Or: "License has expired. Data flow has been stopped.".
- `text`: One of: "License will expire on `${expirationDate}`, no external inputs will be read after that time. Please contact sales@cribl.io to renew your license." Or: "License has expired."

6.9. Scripts

Admins can run scripts (e.g., shell scripts) from within Cribl Edge by configuring and executing them at **Settings** > **Global Settings** > **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!



Settings > Manage Scripts page

The Manage Scripts page provides the following tields:

- **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 Leader Node, but can be **run** only locally from each Worker Node.
- If the Script command is referencing a file (e.g., `420.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.

7. Access Management

Cribl Edge provides a range of access-management features for users with different security requirements. For details, see the following topics:

- Authentication: Authenticating users in Cribl Edge.
- Members and Permissions: Fine-grained access control, available in Cribl Edge 4.2 and later.
- Local Users: Creating and managing users in Cribl Edge's original Role-based model.
- Roles: Managing Roles and Policies to assign to users, in Cribl Edge's original RBAC model.

> Permission- and Role-based access control can be enabled only on distributed deployments (Stream, Edge with an Enterprise license. With other license types and/or single-instance deployments (Stream, Edge), all users will have full administrative privileges.

7.1. 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, navigate to **Settings** > [**Global Settings** >] **Access Management > Authentication** and select **Local**.

You can then manage users through the **Settings** > [**Global Settings** >] **Access Management > Local Users** UI. 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","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","disabled":"false",
"passwd":"Yrt0MOD1w8OzyMYB8WMcEleOtYESMwZw2qIZyTvueOE"}
```

...you'd modify the final key-value pair to something like:

```
{"username":"user","first":"Goat","last":"McGoat","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. From the top nav, select **Manage**.
2. Select the desired Fleet.
3. From the Fleet's submenu, select **Group 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 hybrid Workers include the above **Local Users** options, but Groups for Cribl-managed Workers 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 Settings** >] **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.
- **Idle session TTL**: 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 header**: 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 – e.g., `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 either an Enterprise or a Standard license. They're not supported with a Free license.
>
> 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 Settings** >] **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`.
- **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 Settings** >] **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` (e.g., `localhost:389`).
- **Bind DN**: Distinguished Name of entity to authenticate with LDAP server. E.g., `'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, e.g., `'dc=example,dc=org'`.

- **Username field**: LDAP user search field, e.g., `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, e.g., `dc=example,dc=org`.

- **Group member field**: LDAP group search field, e.g., `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., e.g., `(&(cn=cribl*)(objectclass=group))`.
- **Group name field**: Attribute used in objects' DNs that represents the group name, e.g., `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 (e.g., `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.

7.2. Members and Permissions

Define and manage fine-grained access control across Cribl products and resources

In Cribl Edge 4.2, 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.

> Permission-based access control is available only on distributed deployments (Stream, Edge) with an Enterprise license or Cribl.Cloud plan. With other license/plan types and/or single-instance deployments, all users will have full administrative privileges.
>
> In initial 4.2.x releases, existing Local Users display incorrectly on **Settings** > **Members** pages with `No Access` Permissions. This is a display-only bug: These users' original Roles still function as configured. For details and fix timeline, please see Known Issues.

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 .

Why Move My Cheese?

Members and Permissions enable the finer-grained access control and authorization that many of our users have requested. You can now assign different Cribl users access and capabilities independently at multiple levels:

- Organization.
- Product: Stream, Edge, or Search.
- Group/Fleet.
- Resource: Stream Project, Search Dataset or Dataset Provider, etc.

The benefits?:

- Greater security in protecting access to resources and configuration.
- Greater flexibility in authorizing users on different parts of your Cribl suite (i.e., Organization).

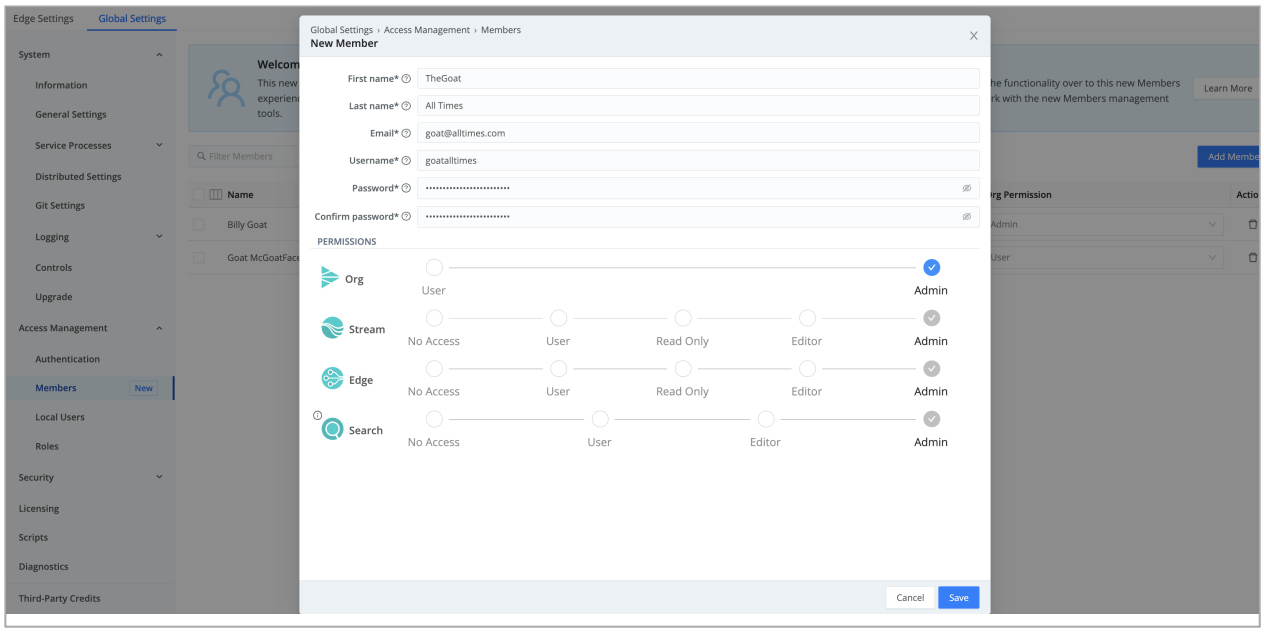
- Ability to segment access and authorization to relevant teams.
- Ability to support a greater diversity of users, given these options.

Cribl plans to eventually retire its earlier RBAC model. But for now, both models are supported, to enable users to experiment and switch over at their own pace. As outlined below, several components of the new versus old model are currently interchangeable.

Organizations

Cribl.Cloud has, from the start, provided the concept of an Organization as a container for the deployment of a whole suite of Cribl products (Stream, Edge, and Cloud-only Search). The Members and Permissions model brings this concept to customer-managed (on-prem) deployments. Here, to access the Organization-level Members UI as an Admin:

1. From Cribl Edge's top nav, select **Settings** > **Global Settings**.
2. From the left nav, select **Members**.
3. Here, you can click an existing Member's row to change their access Permissions or other details, or click **Add Member** to grant a new user access to your Cribl Organization.



Organization > Members/Permissions UI

> For Cribl.Cloud's updated counterpart to this on-prem UI for managing Organization-level Permissions, see Inviting and Managing Other Users.

Members and Local Users

The top half of the **Members** configuration modal is almost identical to Cribl's original Local Users modal. In the current release, Members and Local Users are interchangeable: Members whom you add here can be reconfigured in the traditional **Local Users** UI (using legacy Roles instead of Permissions), and anyone you add in Local Users will also show up here in **Members**.

So let's focus on the the modal's bottom half, which is more interesting – the new graphical **Permissions** management.

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 click along the sliders to assign the following Permission levels.

> For Cribl.Cloud Organization-level Permissions, see our 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.

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.)
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 Settings** > **Members** and click 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:

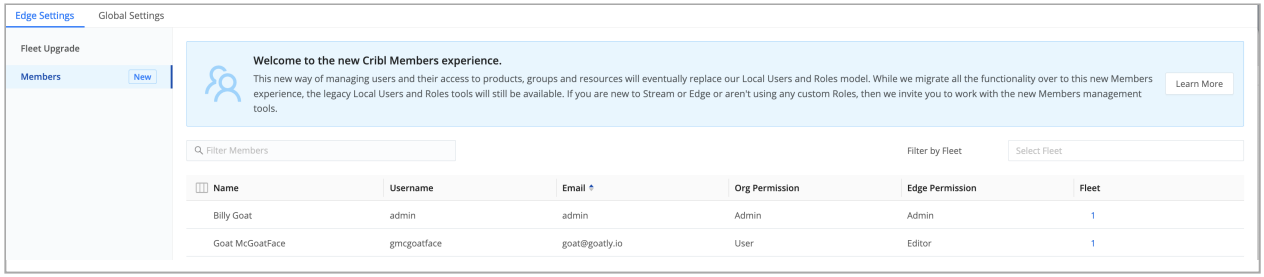
PERMISSION	DESCRIPTION
User	The most basic Permission. Gets the Member into the Fleet, where Admins can then assign them access on individual resources. (Unless Members have a higher Permission at the Product level, they have no initial access to Fleets or resources.)
Read Only	Designed for support personnel. 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

PERMISSION	DESCRIPTION
	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. 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

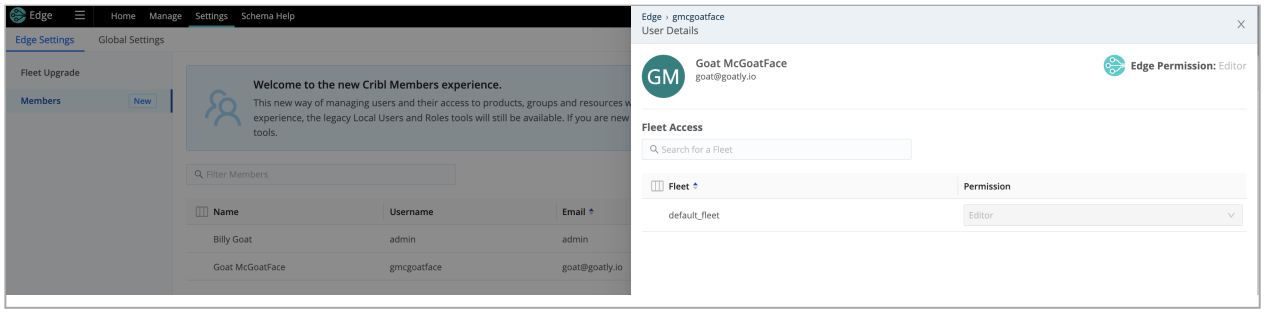
To assign Permissions on a Fleet:

1. Select **Settings** > **Global Settings** > **Members**.
2. Make sure this Member has an **Org Permission** level sufficient to support the Permission you want to assign on the Fleet.
3. From the top-left product switcher, select **Cribl Edge**. Then select (depending on the product) **Settings** > **Stream Settings** > **Members** or **Settings** > **Edge Settings** > **Members**.



Edge Settings > Members

4. Click each applicable Member's row to open their **User Details** drawer.



5. Grant the Member the Permission level you want on this Fleet.

> To change Members' existing Permissions, repeat the same steps above.

Resource-Level Permissions

Certain Cribl products provide Permission-based access to particular resources.

Stream Projects

In v.4.2 and later, Stream Projects and Subscriptions rely entirely on the Members/Permissions access control available in the same versions. 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 now retired.

> **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.

Inheritance

In the current release, 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.

UX Customization

Cribl Edge's UI will be presented differently to users who are assigned Permissions that impose access restrictions. Some controls will be visible but disabled, and some internal-search and log results will be limited, depending on each user's Permissions.

Access to the same objects via Cribl Edge's API and CLI will be similarly filtered, with appropriate error reporting. E.g., if a user tries to commit and deploy changes on a Worker Group/Fleet where they are not authorized, they might receive a CLI error message like this: `git commit-deploy command failed with err: Forbidden`

Legacy Roles and Policies

To facilitate a smooth transition to Cribl's new access-control model – and to provide backward compatibility for customers still using our earlier Roles/Policies model – our pre-4.2 Roles have been supplemented with new counterparts to most of the new Permissions listed above.

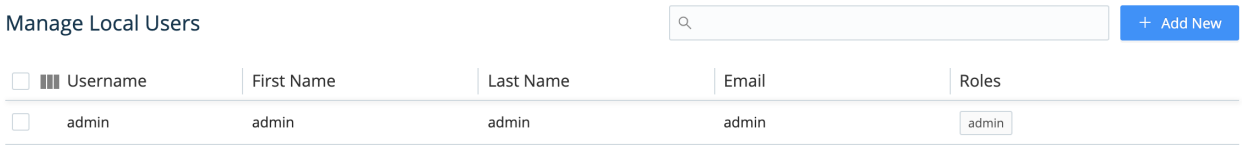
7.3. Local Users

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

On the Leader Node, you manage users by selecting **Settings** > **Global Settings** > **Access Management** > **Local Users**. In single-instance deployments (Stream, Edge), you select **Settings** > **Access Management** > **Local Users**.

The resulting **Local Users** page will initially show only the default `admin` user. You are operating as this user.

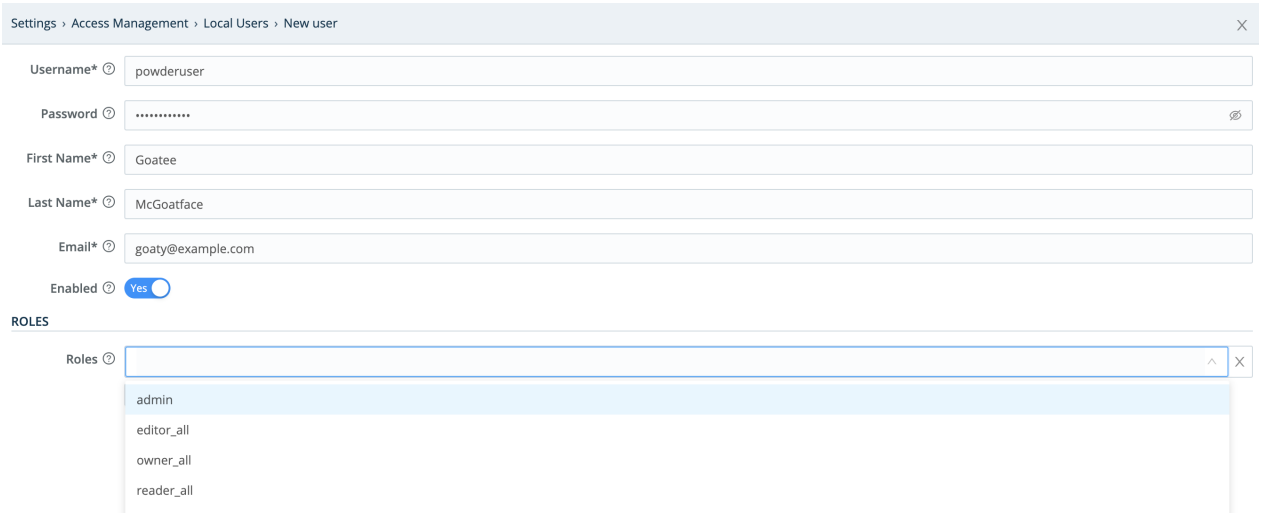


Managing users

To create a new Cribl Edge user, click **New User**. To edit an existing user, click anywhere on its row. With either selection, you will see the modal shown below.

The first few fields are self-explanatory: they establish the user's credentials. Usernames and passwords are case-sensitive.

If you choose to establish or maintain a user's credentials on Cribl Edge, but prevent them from currently logging in, you can toggle the **Enabled** slider to `No`.



Entering and saving a user's credentials

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.

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 an Enterprise license. With other license types and/or single-instance deployments (Edge, Stream), all users will have full administrative privileges.

Click **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 Settings** > **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.

When you've configured (or reconfigured) this user as desired, click **Save**.

By default, Cribl Edge will log out a user upon a change in their assigned Roles. You can defeat this behavior at **Settings** > **Global Settings** > **General Settings > API Server Settings > Advanced > Logout on roles change.**

7.4. Roles

Define and manage access-control Roles and Policies

Cribl Edge offers role-based access control (RBAC) to serve 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.

> Role-based access control is available only on distributed deployments (Stream, Edge) with an Enterprise license or Cribl.Cloud plan. With other license/plan types and/or single-instance deployments, all users will have full administrative privileges.

Roles, Meet Permissions

In Cribl Edge 4.2 and later, the Roles/Policies model described on this page exists in parallel with a new, more flexible Members/Permissions model, which will eventually replace it. To provide cross-compatibility, we have added several new Default Roles and Default Policies that are counterparts to new Permissions.

As noted below, these cross-compatible Roles/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.

> **Known Issue**
>
> In initial 4.2.x releases, existing Local Users display incorrectly with `No Access` Permissions on **Settings** > **Members** pages. This is a display-only bug: These users' Roles still function as originally configured. For details and fix timeline, 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.

How Cribl Edge RBAC Works

Cribl Edge RBAC is designed to grant arbitrary permissions over objects, attributes, and actions at arbitrary levels.

> In Cribl Edge v.2.4.x through 4.1.x, Roles are customizable only down to the Worker Group/Fleet level. E.g., you can grant Edit permission on Worker Group/Fleet `WG1` to User A and User B, but cannot grant them finer-grained permissions on child objects such as Pipelines, Routes, etc.

Cribl Edge's UI will be presented differently to users who are assigned Roles that impose access restrictions. Controls will be visible but disabled, and search and log results will be limited, depending on each user's permissions.

Access to the same objects via Cribl Edge's API and CLI will be similarly filtered, with appropriate error reporting. E.g., if a user tries to commit and deploy changes on a Worker Group/Fleet where they are not authorized, they might receive a CLI error message like this: `git commit-deploy command failed with err: Forbidden`

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.

> Where we list an equivalent Permission (Cribl Edge 4.2 and later), see Members and Permissions for details.

NAME	DESCRIPTION
admin	Superusers – authorized to do anything and everything in the system.
owner_all	Read/write access to (and Deploy permission on) all Worker Groups/Fleets.
editor_all	Read/write access to all Worker Groups/Fleets.
reader_all	Read-only access to all Worker Groups/Fleets.
collect_all	Ability to create, configure, and run Collection jobs on all Worker Groups/Fleets.
gitops	Ability to sync the Cribl Edge deployment to a remote Git repository.
notification_admin	Read/write access to all Notifications.
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.
project_user	Read/write access to the simplified Stream Projects UI and related data Subscriptions. **Deprecated as of v.4.2:** Assign the new Project-level **Editor** Permission instead.
stream_user	Provides compatibility with 4.2+ Stream **User** Permission.
stream_reader	Provides compatibility with 4.2+ Stream **Read Only** Permission.
stream_editor	Provides compatibility with 4.2+ Stream **Editor** Permission.
stream_admin	Provides compatibility with 4.2+ Stream **Admin** Permission.
edge_user	Provides compatibility with 4.2+ Edge **User** Permission.
edge_reader	Provides compatibility with 4.2+ Edge **Read Only** Permission.
edge_editor	Provides compatibility with 4.2+ Edge **Editor** Permission.

NAME	DESCRIPTION
edge_admin	Provides compatibility with 4.2+ Edge **Admin** Permission.
search_user	Provides compatibility with 4.2+ Search **User** Permission.
search_editor	Provides compatibility with 4.2+ Search **Editor** Permission.
search_admin	Provides compatibility with 4.2+ Search **Admin** Permission.

Cribl **strongly recommends** that you do not edit or delete these default Roles. However, you can readily clone them (see the **Clone Role** button below), and modify the duplicates to meet your needs.

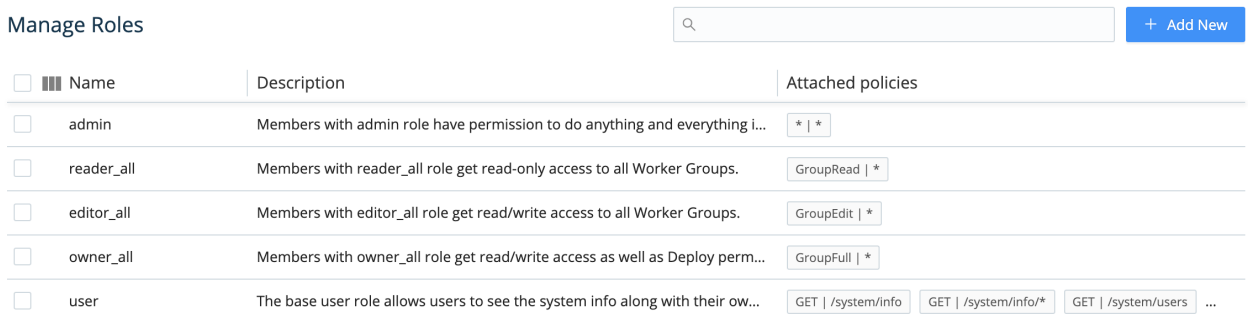
Initial Installation or Upgrade

When you first install Cribl Edge with the prerequisites to enable RBAC (Enterprise license and distributed deployment), 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.

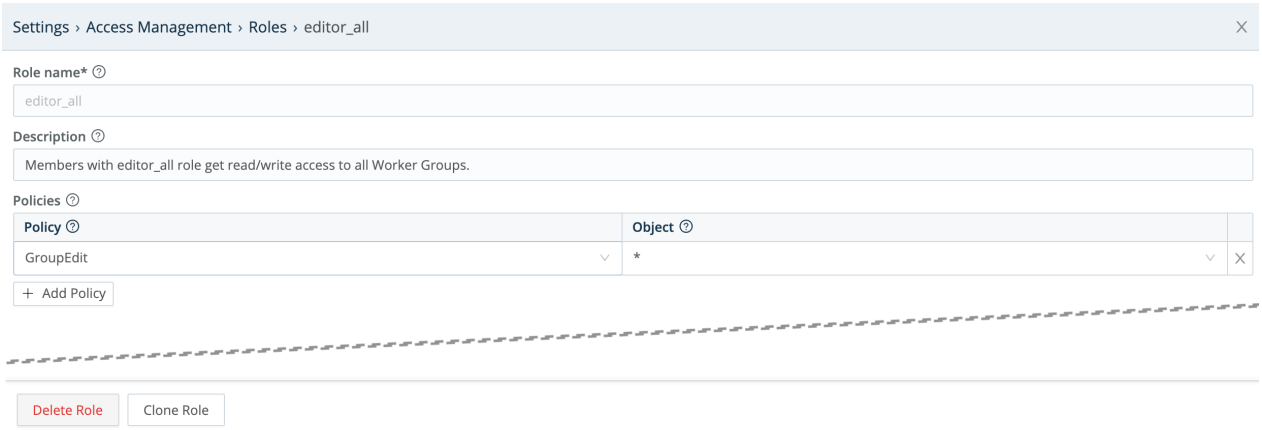
Adding and Modifying Roles

In a distributed environment, you manage Roles at the Leader level, for the entire deployment. On the Leader Node, select **Settings** > **Global Settings** > **Access Management** > **Roles**.



Manage Roles page

To add a new Role, click **New Role** at the upper right. To edit an existing Role, click anywhere on its row. Here again, either way, the resulting modal offers basically the same options.



Add/edit Role modal

The options at the modal's top and bottom are nearly self-explanatory:

Role name: Unique name for this Role.

Description: Optional free-text description.

Delete Role: And...it's gone. (But first, there's a confirmation prompt. Also, you cannot delete a Role assigned to an active user.)

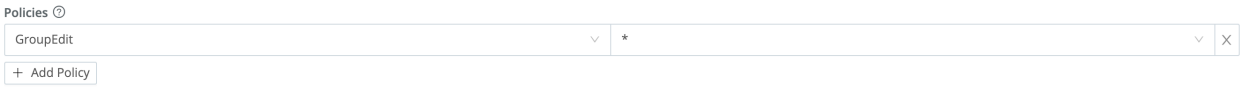
Clone Role: Opens a **New role** version of the modal, duplicating the **Description** and **Policies** of the Role you started with.

The modal's central **Policies** section (described below) is its real working area.

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 Worker Groups/Fleets (as indicated by the `*` wildcard).



Policies on the left, objects on the right

To add a new Policy to a Role:

1. Click **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, click its close box 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:

NAME	DESCRIPTION
`GroupRead`	The most basic Worker Group/Fleet-level permission. Enables users to view a Worker Group/Fleet and its configuration, but not modify or delete the config. Counterpart to 4.2+ Fleet-level **Read Only** Permission.
`GroupEdit`	Building on `GroupRead`, grants the ability to also change and commit a Worker Group/Fleet's configuration. Counterpart to 4.2+ Fleet-level **Editor** Permission.
`GroupFull`	Building on `GroupEdit`, grants the ability to also deploy a Worker Group/Fleet. Counterpart to 4.2+ Fleet-level **Admin** Permission.
`GroupCollect`	Grants the ability to create, configure, and run Collectors on a Worker Group.
`GroupUser`	Provides compatibility with 4.2+ Fleet-level **User** Permission.
`ProjectMaintain`	Provides compatibility with 4.2+ Project-level **Maintainer** Permission.
`ProjectRead`	Provides compatibility with 4.2+ Project-level **Read Only** Permission.
`ProjectEdit`	Provides compatibility with 4.2+ Project-level **Editor** Permission.
`Product`	[Internal building block for Product-specific Policies. Do not add directly to Roles.]
`BaseProductUser`	[Internal building block for Product-specific User Policies. Do not add directly to Roles.]
`ProductUser`	Provides compatibility with 4.2+ Product-level **User** Permission.

NAME	DESCRIPTION
`LimitedProductUser`	Similar to `ProductUser`, but omits the ability to read or act on all the endpoints within a Fleet.
`ProductAdmin`	Provides compatibility with 4.2+ Product-level **Admin** Permission.
`DatasetMaintain`	Provides compatibility with 4.2+ Search datasets **Maintainer** Permission.
`DatasetRead`	Provides compatibility with 4.2+ Search datasets **Read Only** Permission.
`DatasetProviderMaintain`	Provides compatibility with 4.2+ Search dataset providers **Maintainer** Permission.
`DatasetProviderRead`	Provides compatibility with 4.2+ Search dataset providers **Read Only** Permission.
`SearchBase`	Similar to `SearchUser`: Can view Search resources shared with the Member. [Internal building block for Search Policies. Do not add directly to Roles.]
`SearchUser`	Provides compatibility with 4.2+ Search Product-level **User** Permission.
`SearchMaintainer`	Provides compatibility with 4.2+ Search Product-level **Editor** Permission.
`MaintainBase`	[Internal building block for `Maintain` Policies. Do not add directly to Roles.]
`*` (wildcard)	Grants **all** permissions on associated objects.

> **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. (Remember that in v. 2.4, the objects available for selection are specific Worker Groups/Fleets, or a wildcard representing all Worker Groups/Fleets.) For example:

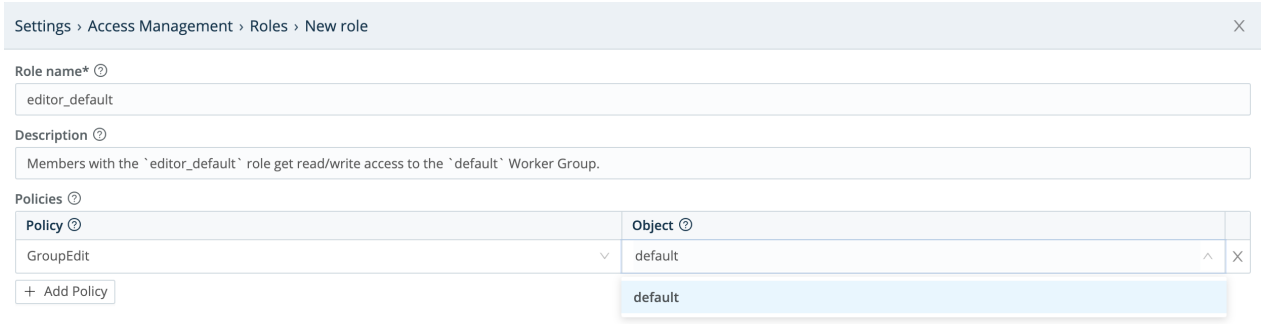
- `Worker Group <id>`
- `NewGroup2`
- `default` (Worker Group)
- `*` (all Worker Groups)

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 Worker Group/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` Worker Group/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 Worker Group/Fleet of your own.

Roles and Users

Once you've defined a Role, you can associate it with Cribl Edge users. On the Leader Node, select **Settings** > **Global Settings** > **Access Management** > **Local 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.

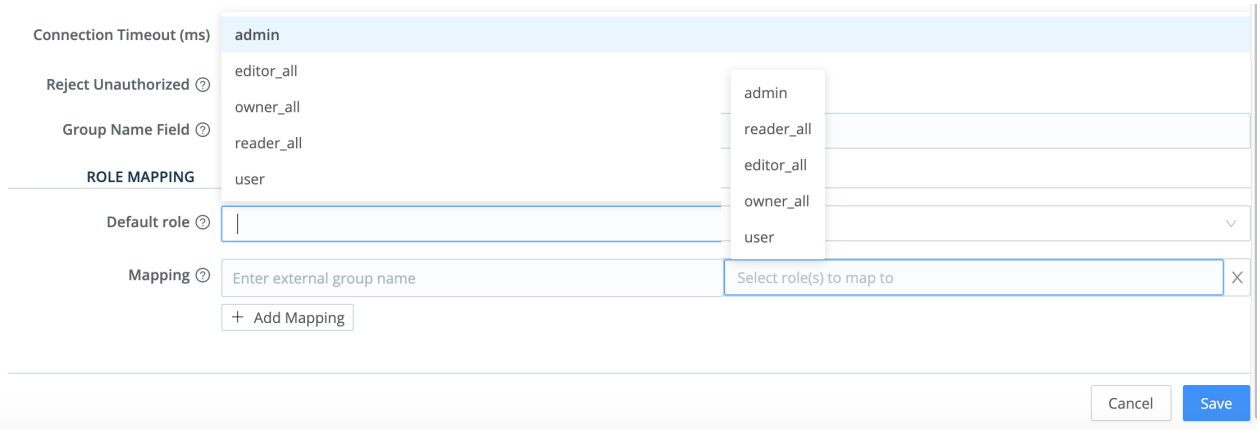
By default, Cribl Edge will log out a user upon a change in their assigned Roles. You can defeat this behavior at **Settings** > **Global Settings** > **General Settings > API Server Settings > Advanced > Logout on roles change.**

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:

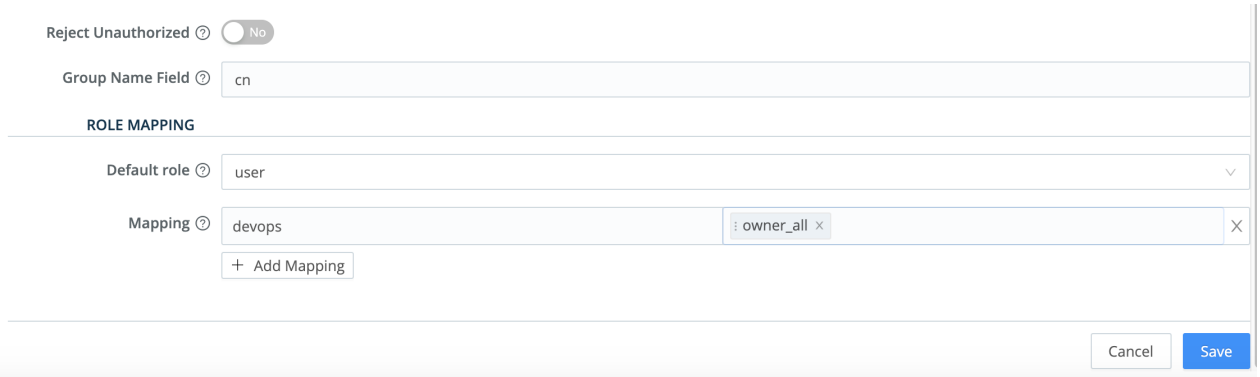
1. Select **Settings** > **Global Settings** > **Access Management** > **Authentication**.
2. From the **Type** drop-down, select **LDAP**, **Splunk**, or **OpenID Connect**, according to your needs.
3. On the resulting **Authentication Settings** page, configure your identity provider's connection and other basics. (For configuration details, see the appropriate Authentication section.)
4. 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.
5. 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).
6. To map more user groups, click **Add Mapping**.
7. When your configuration is complete, click **Save**.

Here's a composite showing the built-in Roles available on both the **Default role** and the **Mapping** drop-downs:



Mapping external user groups to Cribl Edge Roles

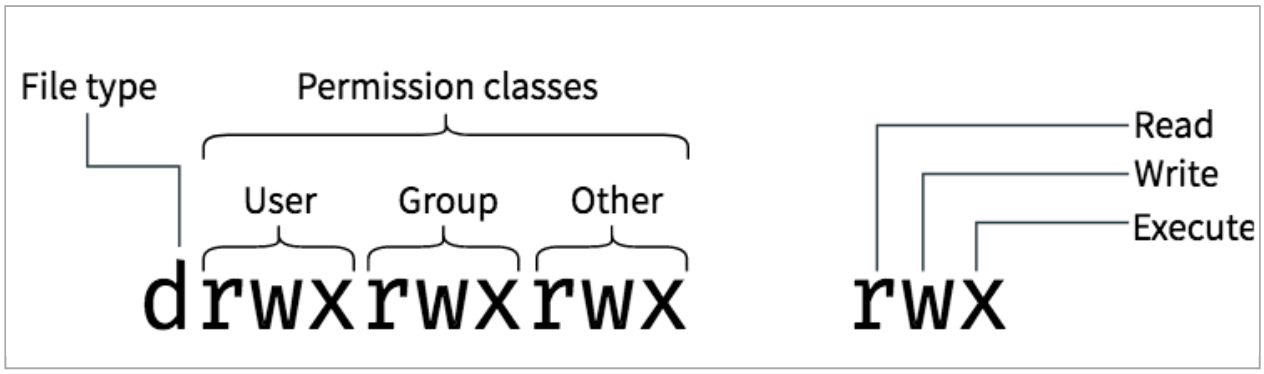
And here, we've set a conservative **Default Role** and one explicit **Mapping**:



External user groups mapped to Cribl Edge Roles

7.5. 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. Securing

8.1. Securing the Leader's Auth Token

Cribl **strongly recommends** that when you first install Cribl Edge, you change the Leader's auth token value from the default `criblmaster` to a unique secure value.

This applies whether you configure your Leader via the UI, or via a Docker Compose file.

Changing the Default Auth Token Value

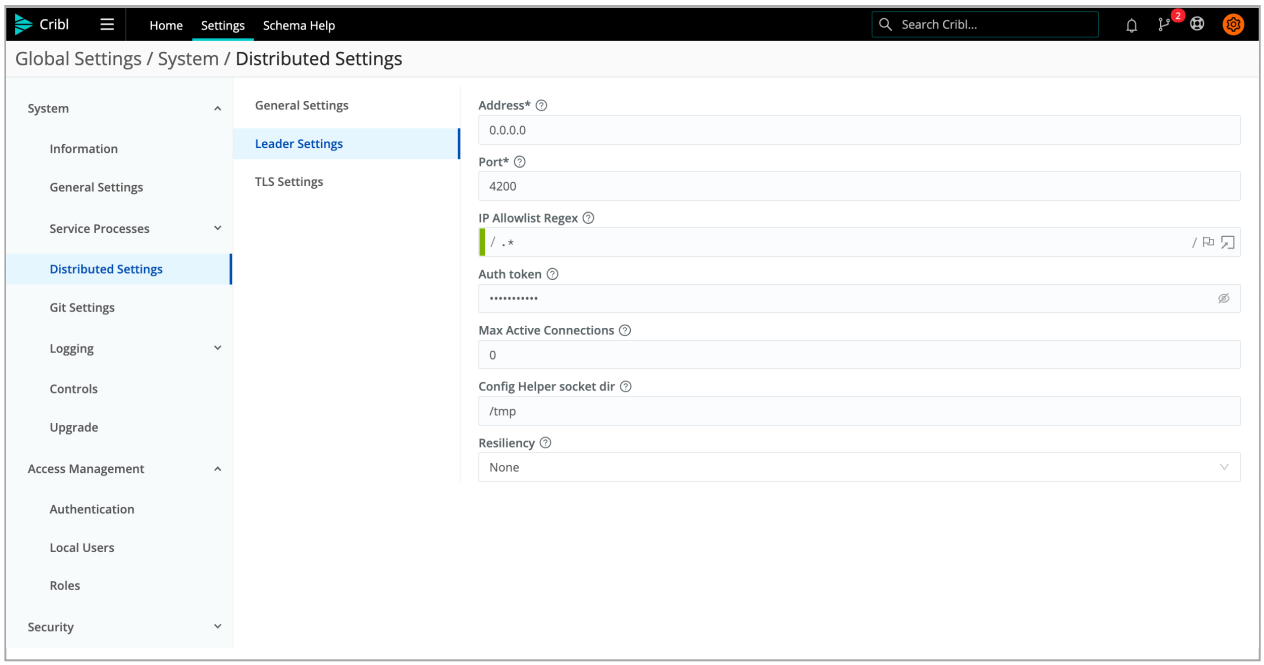
Create a unique, strongly secure token value. Cribl recommends creating a string that contains at least 14 characters and includes uppercase letters, lowercase letters, and numbers.

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, paste it into the **Auth token** field in the UI, or in the appropriate place in your Docker Compose file (if applicable).

In the UI, you'll find the **Auth token** setting under **Global Settings** > **System** > **Distributed Settings** > **Leader Settings**, as shown in the screenshot below.



Distributed > Leader Settings

8.2. Securing Cribl Edge (TLS/SSL, Certs, Keys)

You can secure Cribl Edge access and traffic using various combinations of SSL (Secure Sockets Layer), TLS (Transport Layer Security), custom HTTP headers, and internal or external KMS (Key Management Service) options.

> In a single-instance deployment (Stream, Edge), wherever this page refers to a Fleet, just follow the Cribl Edge top nav's **Manage** link.

Secure Access to Edge Nodes' UI

A best practice in enterprise distributed deployments, this prevents direct browser access to Edge Nodes' UI.

> Cribl recommends that you first enable the Leader's UI access to Edge Nodes (Stream, Edge). This way, admins will still be able to tunnel through from the Leader to any Edge Node's UI. This is also a prerequisite for Connecting Workers to the Leader Securely.

1. Select a Fleet.
2. Open **Fleet Settings** (top right).
3. Under **General Settings** > **API Server Settings**, click **Advanced**.
4. Toggle the **Local UI Access** slider to `No`.
5. Click **Save**.

SSL Certificate Configuration

You can secure Cribl Edge's API and UI access by configuring SSL. Do this on the Leader, to secure Edge Nodes' inbound communications.

You can use your own certs 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 can configure the cert via **Settings** > **Global Settings** > **Security > Certificates**. You can configure the **key** via:

- **Settings** > **Global Settings** > **Security > Encryption Keys** single-instance deployments (Edge, Stream), or
- **Manage** > **Groups** > `<group-name>` > **Group Settings** > **Security** > **Encryption Keys** distributed deployments (Edge, Stream).

Alternatively, you can edit the `local/cribl.yml` file's `api` section to directly set the `privKeyPath` and `certPath` attributes. For example:

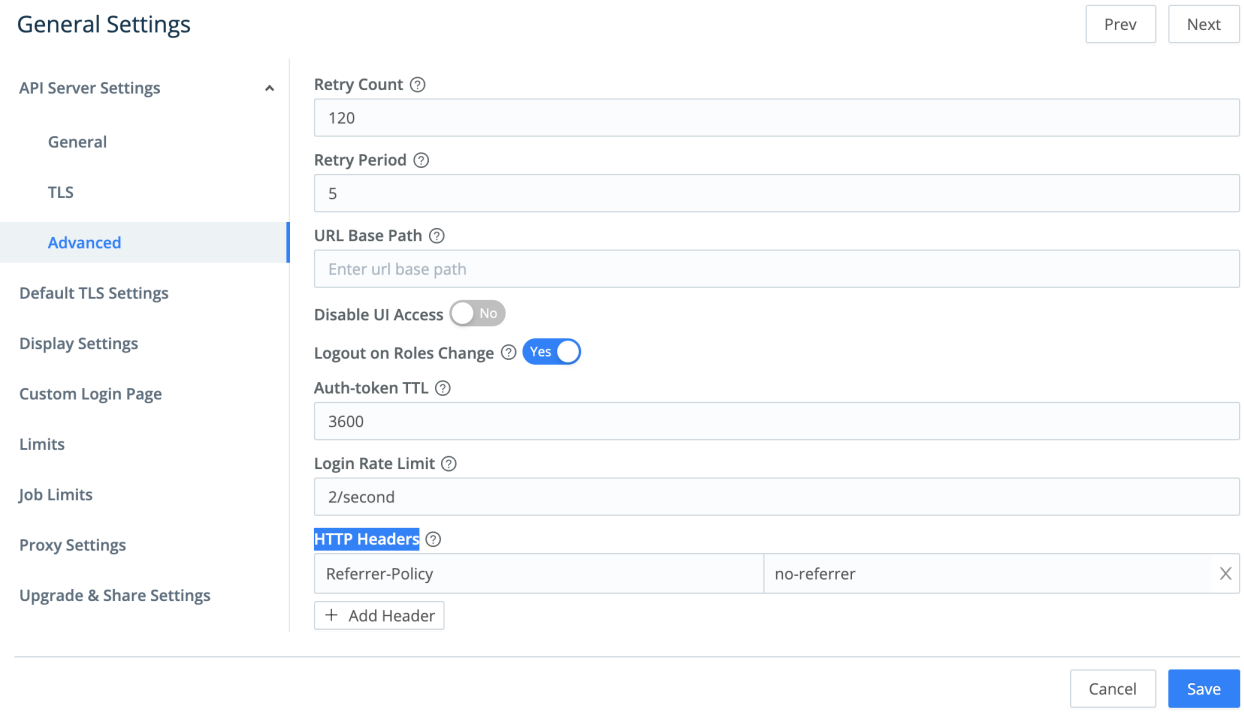
cribl.yml

```
api:
  host: 0.0.0.0
  port: 9000
  disabled : false
  ssl:
    disabled: false
    privKeyPath: /path/to/myKey.pem
    certPath: /path/to/myCert.pem
...
```

> See Securing Communications for details about using this certificate and key to secure communications on, and among, your Cribl Edge Leader and Edge Nodes.

Custom HTTP Headers

You can encode custom, security-related HTTP headers, as needed. As shown in the examples below, you specify these at **Settings** > **Global Settings** > **General Settings** > **API Server Settings** > **Advanced** > **HTTP Headers**. Click **Add Header** to display extra rows for new key-value pairs.



Custom HTTP headers

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

Default TLS Settings (Cyphers, Etc.)

You can configure advanced, system-wide TLS settings – minimum and maximum TLS versions, default cypher lists, and ECDH curve names. Select **Settings** > **Global Settings** > **System > General Settings > Default TLS Settings**.

Here, in Cribl Edge's **Default cypher list** field, you can specify between one and all of the following supported cyphers:

- `TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384`
- `TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256`
- `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`
- `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`

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.

> In Cribl Edge 4.1 and later, 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 later), 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 Restoring Unencrypted Private Keys below.

Accessing 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 this key's purpose.

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` (Cribl Edge's internal Key Management Service).

Key Class: Classes are arbitrary collections of keys that you can map to different levels of access control. For details, see Encryption. 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.

Restoring Unencrypted Private Keys

If you need to roll back from Cribl Edge 4.1 or later to an earlier version, follow this procedure to restore the unencrypted private key files that you earlier backed up.

1. Stop Cribl Edge 4.1 (or later) on hosts – the Leader and all Edge Nodes.
2. Untar the older Cribl Edge version (e.g., 4.0.4) 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.

Secrets

With Cribl Edge's secrets store, you can centrally manage secrets that Cribl Edge instances use to authenticate on integrated services. Use this UI section to create and update authorization tokens, username/password combinations, and API-key/secret-key combinations for reuse across the application.

Accessing 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 resulting **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: See below for this second field's options, some of which expose additional controls.

Description: Optionally, enter a description summarizing this secret's purpose.

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 supported on Cribl Edge's AWS-based Sources, Collectors, and

Destinations, and on our Google Cloud Storage Destination.

Username with password: Exposes **Username** and **Password** fields, which you fill to retrieve the secret using Basic Authentication.

CA Certificates and Environment Variables

Where Cribl Edge Sources and Destinations support TLS, each Source's or Destination's configuration provides a **CA Certificate Path** field where you can point to corresponding Certificate Authority (CA) `.pem` file(s). However, you can also use environment variables to manage CAs globally. Here are some common scenarios:

1. **How do I add a set of trusted root CAs to the list of trusted CAs that Cribl Edge trusts?**
   Set this environment variable in each Edge Node's environment (e.g., in its systemd unit file): `NODE_EXTRA_CA_CERTS=/path/to/file_with_certs.pem`. For details, see the nodejs docs.

2. **How do I make Cribl Edge trust all TLS certificates presented by any server it connects to?**
   Set this environment variable: `NODE_TLS_REJECT_UNAUTHORIZED=0` – for details, see the nodejs docs.

8.2.1. KMS Configuration

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 requires an Enterprise or Standard license.

> 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 Settings** > **Security > KMS**. 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** 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 Settings** > **Worker Processes**.
> 2. In the list of processes, locate any process with a Role of `CONFIG_HELPER`.
> 3. Click 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

Setting the **KMS provider** drop-down to `HashiCorp Vault` exposes the following configuration options:

KMS Settings

Vault URL: Enter the Vault server's URL (e.g., `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, e.g. 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, e.g., 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.

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, e.g.: `<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

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.

8.3. Securing Communications

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.

Importing 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** > **Global Settings** > **Security** > **Certificates** > **New Certificates** modal.
2. Open your TLS/SSL certificate file. (The self-signed certificate example 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.

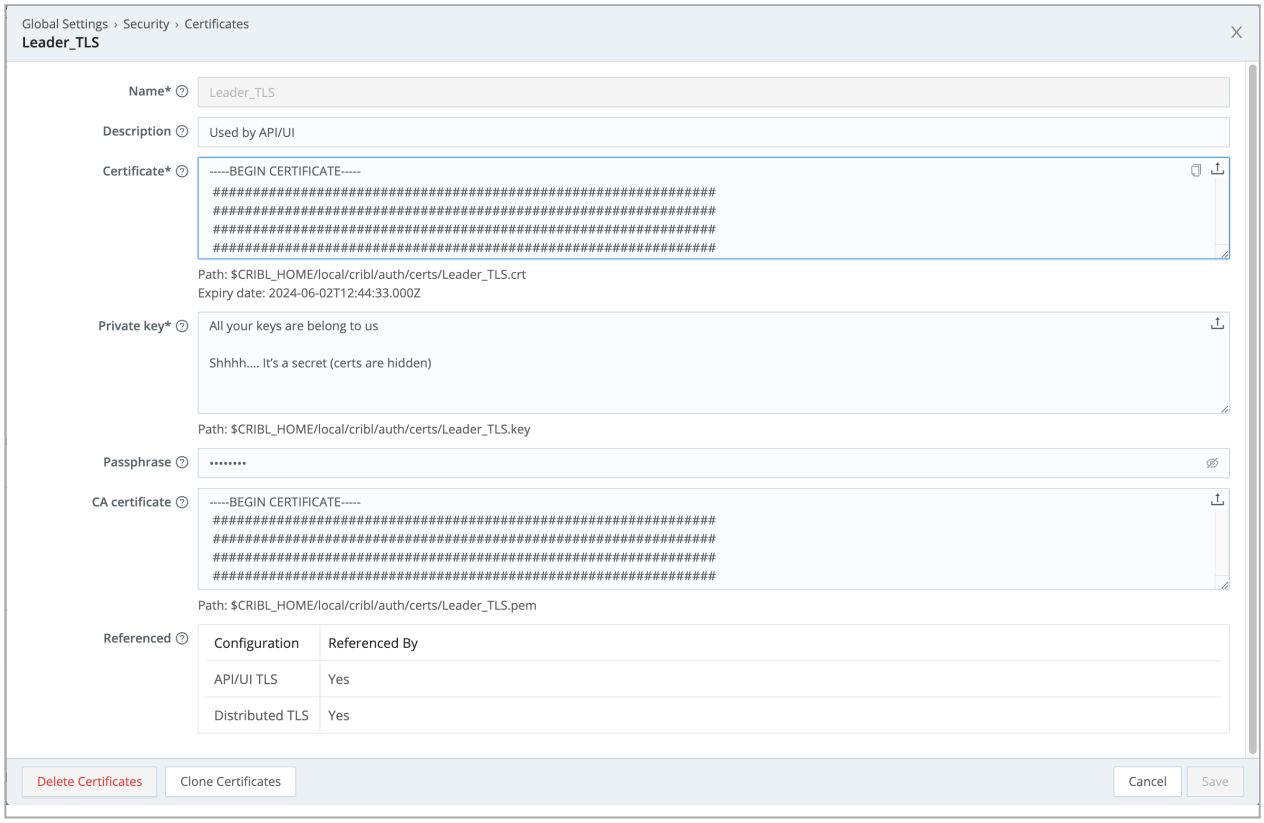
5. Open your private key file. (The self-signed certificate example used the placeholder name `myKey.pem`.)
6. Copy the file's contents to your clipboard.
7. Paste the clipboard contents into the same Leader modal's **Private key** field.

> 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 – e.g., a downloaded Splunk Cloud certificate – import the chain into the **CA certificate** field before saving the cert. For details, see Obtain the Certificate Chain (TLS/SSL).



Leader's certificate modal, populated

Connecting 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.

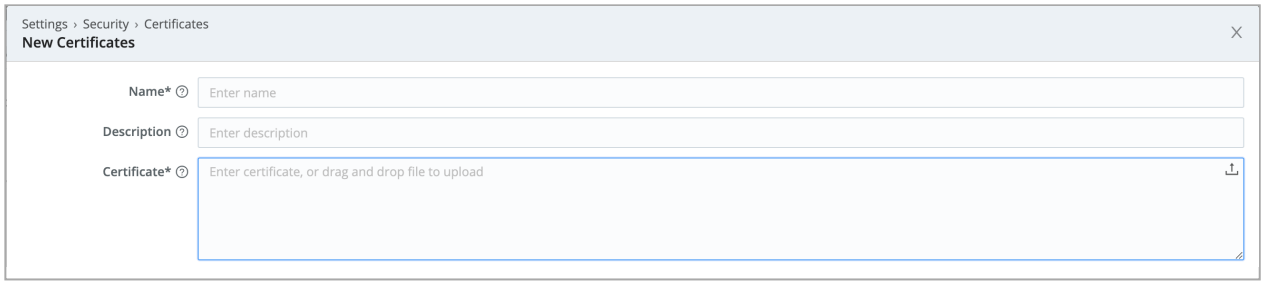
Using the UI

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 cert.)
2. Select **Manage**, then select the Fleet you want to configure.
3. Select **Group Settings** or **Fleet Settings** (upper right).
4. From the left nav, select **Security** > **Certificates** > **TLS**.
5. Click **Add Certificate**.
6. In the resulting **New Certificates** modal, add the cert's contents to the **Certificate** field.
   (You can paste the cert 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 cert.
10. Repeat the preceding steps on each Fleet.

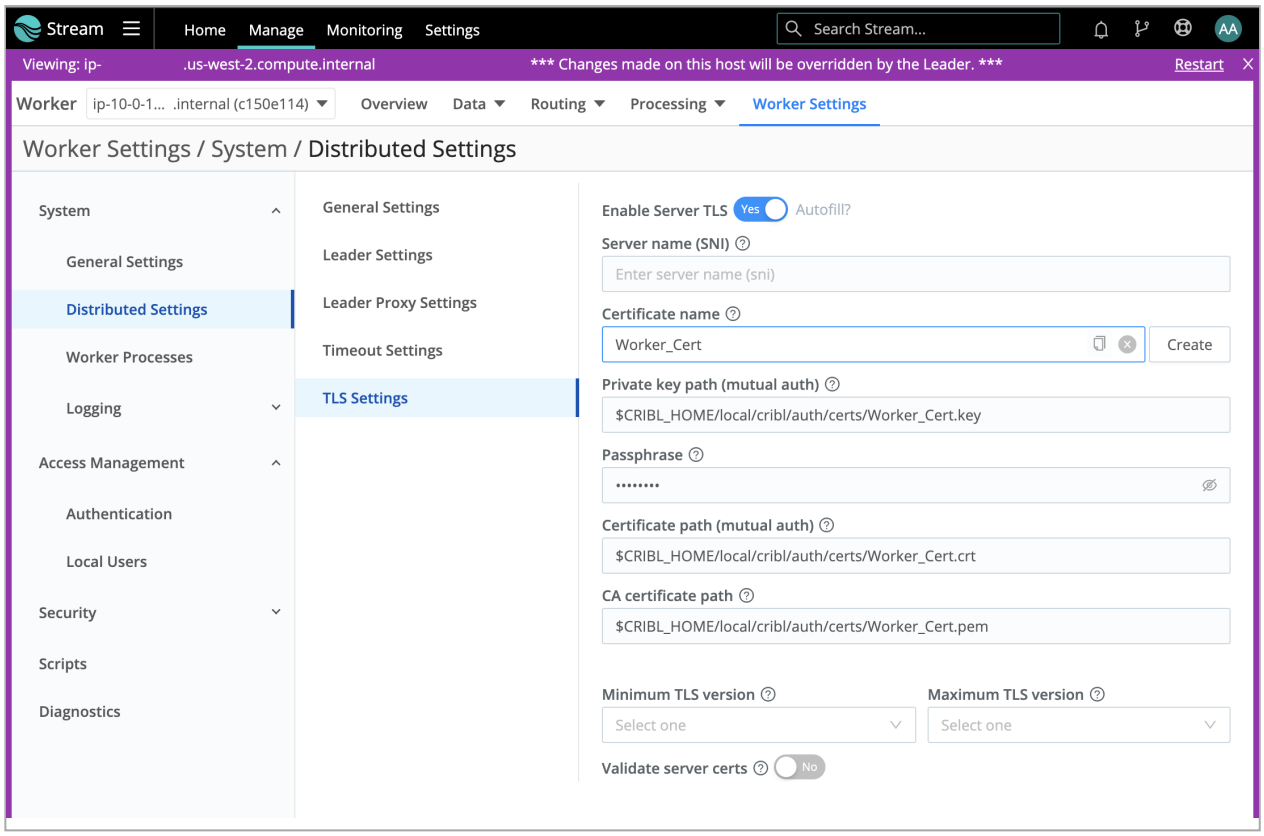


Group-level certificates are configured like the Leader's cert

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 a 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 Worker's/Edge Node's UI, from the Leader

Leader Setup

Next, return to the Leader's UI:

1. Select **Settings** > **Global Settings**> **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 cert 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.

Using 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: criblmaster
      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.

Using 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
```

Configuring TLS Mutual Authentication

Once you have configured the Leader for encrypted TLS communication, you can implement client certificate exchange to enable mutual authentication. 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.

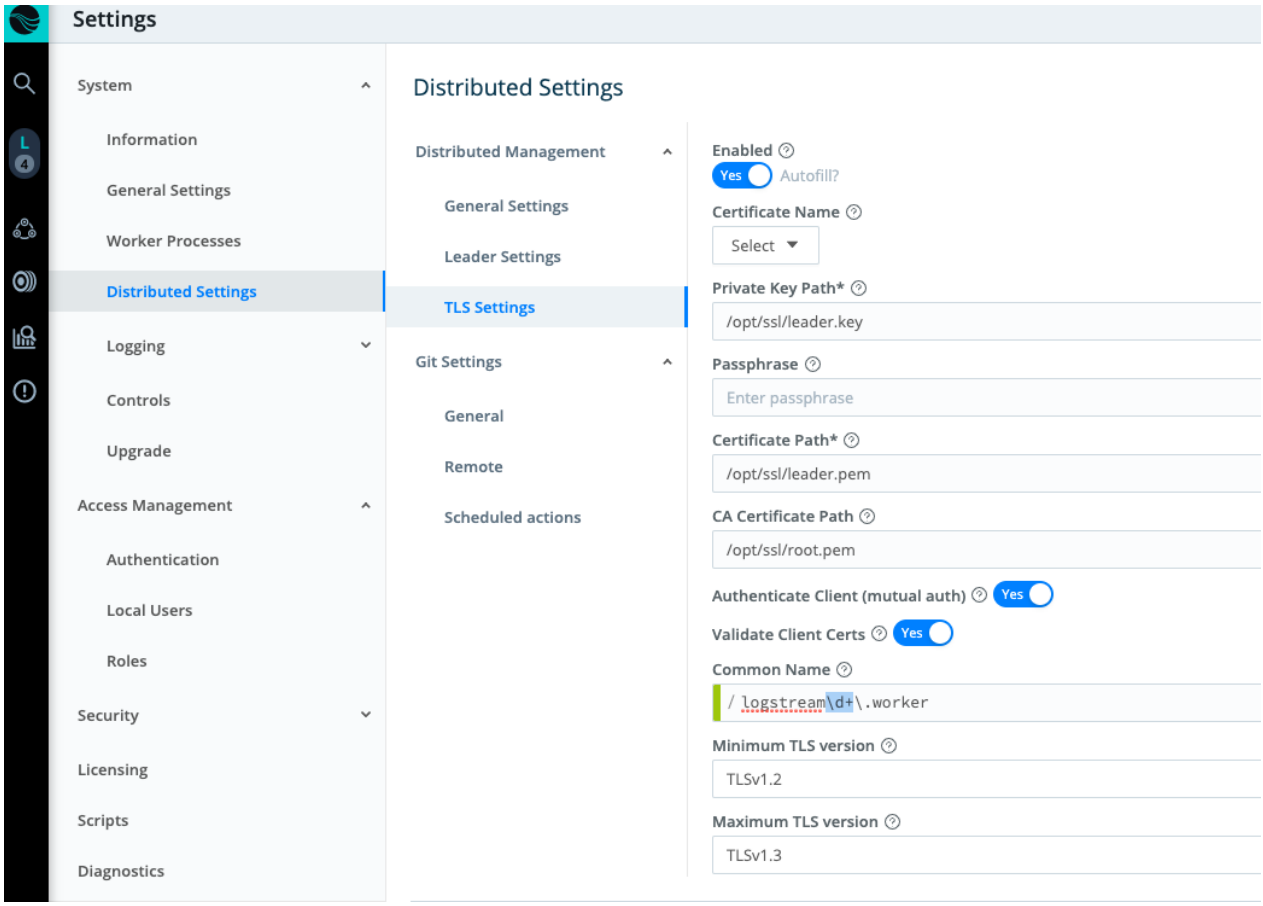
Limitations on TLS Mutual Auth

When configuring TLS mutual 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.

Using the UI

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 TLS **mutual** 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: criblmaster
    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 TLS mutual 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/certs/worker.key&tls.certPath=/path/to/certs/workers.pem&tls.c
```

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 Edge Worker-to-Leader Communications.

Setting Authentication on Sources/Destinations

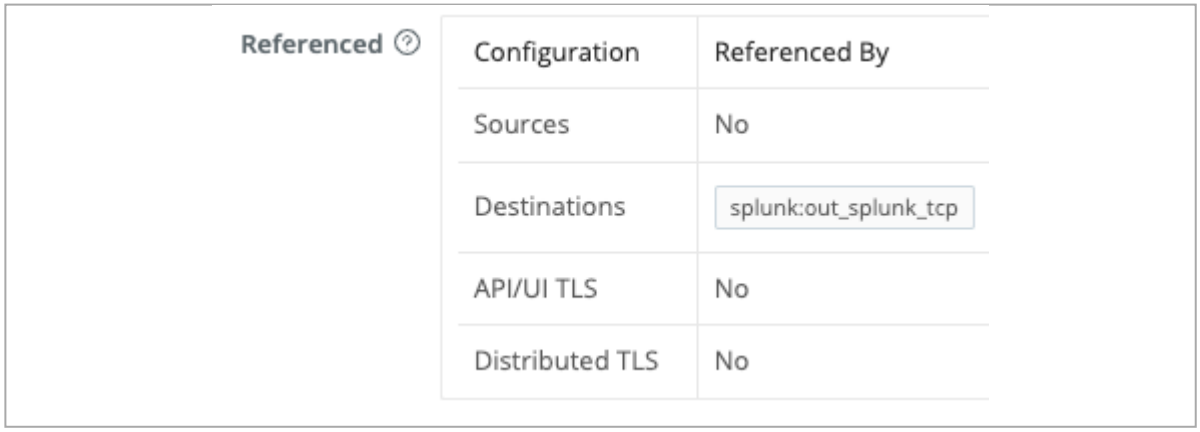
You can use certificates to authenticate Cribl Edge to external data senders and receivers. You configure this at the Group level, as follows:

1. Select a Group.

> As an alternative to the preconfiguration in steps 2–6, you can import a certificate on the fly, using the Source's or Destination's **Create** button. See this section's final steps below.

2. Open **Group Settings** (top right) > **Security** > **Certificates**.
3. Select **New Certificates**.
4. Paste or upload `.pem` files, as in Setting Up the Encrypted Channel.
5. Supply a **Passphrase** and/or **CA certificate**, if required by your integration partner's certificate.
6. Click **Save**.
7. Open your relevant Source's or Destination's config modal.
8. Select **TLS Settings (Client Side)** or **TLS Settings (Server Side)**, depending on the integration.
9. Slide **Enabled** on.
10. In the **Certificate name** drop-down, select a cert that you've preconfigured for this integration. This will auto-populate the corresponding fields here.
11. If you're creating a certificate on the fly, click the **Create** button beside **Certificate name**.
12. Click **Save**.

> Cribl Edge will create new certificates at the same Group level that you're configuring. You can verify this at the **Create new certificate** modal's bottom, by making that the **Referenced** table includes rows for poulated for the appropriate **Sources** and **Destinations**.



Group-level certificate modal

Securing 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 cert we created in Securing Cribl Edge:

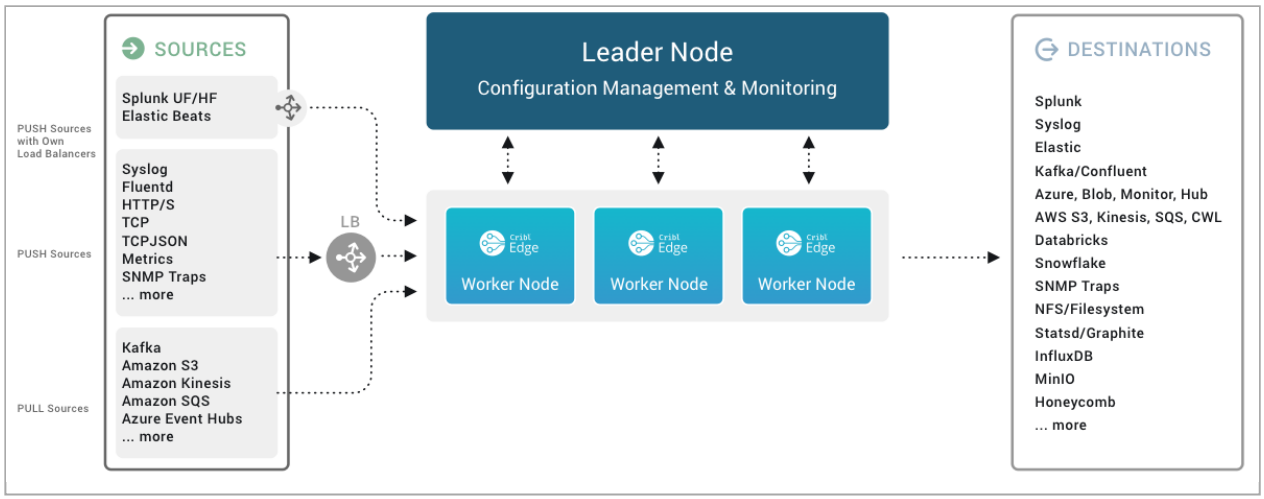
1. Navigate to **Settings** > **Global Settings** > **General Settings** > **API Server Settings** > **TLS**.
2. Slide the toggle to **Enabled**.
3. From the **Certificate Name** drop-down, select a cert you've previously imported. This will populate the corresponding fields here.
4. Click **Save**.

> 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. Sources

Each Cribl Edge **Source** is a configuration that enables Edge nodes to collect or receive observability data – logs, metrics, application data, etc. – 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.

Edge's UI 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. E.g., 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; or 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

- Kubernetes Logs
- Kubernetes Metrics
- System Metrics
- System State
- Windows Metrics

PUSH Sources

Supported data Sources that **send** to Cribl Edge:

- Amazon Kinesis Firehose
- Datadog Agent
- Elasticsearch API
- Grafana
- HTTP/S (Bulk API)
- Loki
- Metrics
- 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

Supported Sources that Cribl Edge **fetches** data from:

- Prometheus Edge Scraper
- Prometheus Scraper (Deprecated)

- Windows Event Logs

Configuring and Managing Sources

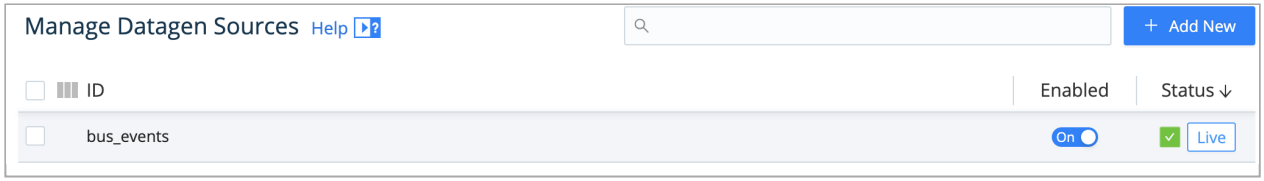
For each Source *type*, you can create multiple definitions, depending on your requirements.

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**.

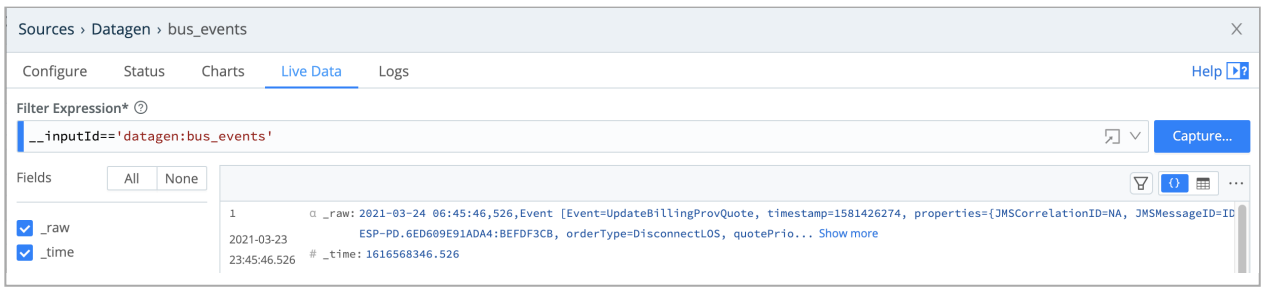
Capturing Source Data

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.



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.



Source modal > Live Data tab

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:

- **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

> Cribl University offers a course titled Collecting Data in Edge that provides an illustrated overview. To follow the direct course link, first log into your Cribl University account. (To create an account, click 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.

9.1. System

9.1.1. Exec

The Exec Source enables you to periodically execute a command and collect its `stdout` output. This is typically used in cases when Cribl Edge cannot accomplish collection with native Collectors or other Sources.

Type: **System and Internal** | 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.

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.

Configuring an Exec Source

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Source** at left. From the resulting drawer's tiles, select [**System and Internal** >] **Exec**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). From the resulting page's tiles or left nav, select [**System and Internal** >] **Exec**. Next, click **New Source** to open a **New Source** modal that provides the options below.

General Settings

Enabled: Defaults to `Yes`.

Input ID: Enter a unique name to identify this Exec Source definition.

Command: Command to execute; supports Bourne shell syntax.

Schedule type: Use the buttons to select either `Interval` or `Cron`. 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.)

Optional Settings

Max retries: Maximum number of retry attempts in the event that the command fails.

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Edge. Use a tab or hard return between (arbitrary) tag names.

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.

9.1.2. File Monitor

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. This Source is available on Stream and Edge, but not on Cribl.Cloud.

> Type: **System and Internal** | 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.

> **Known Issue: Memory Leak**
>
> The 4.2.0/4.2.1 File Monitor Source, when run in Manual Discovery mode, causes a memory leak that eventually causes Edge Nodes to crash. For details, fix timeline, and workaround, please see Known Issues.

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`.)

Configuring a File Monitor Source

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Source** at left. From the resulting drawer's tiles, select [**System and Internal** >] **File Monitor**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). From the resulting page's tiles or left nav, select [**System and Internal** >] **File Monitor**. Next, click **New Source** to open a **New Source** modal that provides the options below.

General Settings

Enabled: Toggle this slider to `Yes` to enable the Source.

Input ID: Enter a unique name to identify this File Monitor Source definition.

Discovery Mode: Use the buttons to select one of these 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** (i.e., a directory) that you specify, down to the **Max depth**. The defaults are:
  - **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.

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`.

> 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.

Max age duration: Optionally, specify a maximum age of files to monitor. This Source will filter events with timestamps newer than the configured duration. (Where files don't have a parsable timestamp, it will set their events' timestamp to `Now`.) Enter a duration in a format like `30s`, `4h`, `3d`, or `1w`. Defaults to an empty field, which applies no age filters.

Check file modification times: If toggled to `Yes`, this Source will skip files with modification times older than the configured **Max age duration**.

> When you set a **Max age duration** 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`, then upon Cribl Edge's next startup, this Source will skip to the end of new files. (It will run discovery once, adding any newly discovered files to the state store as if they had already been consumed.) After that initial run, when this Source next discovers new files, it will read those new files from the head.

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.

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Edge. Use a tab or hard return between (arbitrary) tag names.

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).

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** (e.g., `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.

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/*`

Example: Excluding the System's Own Logs

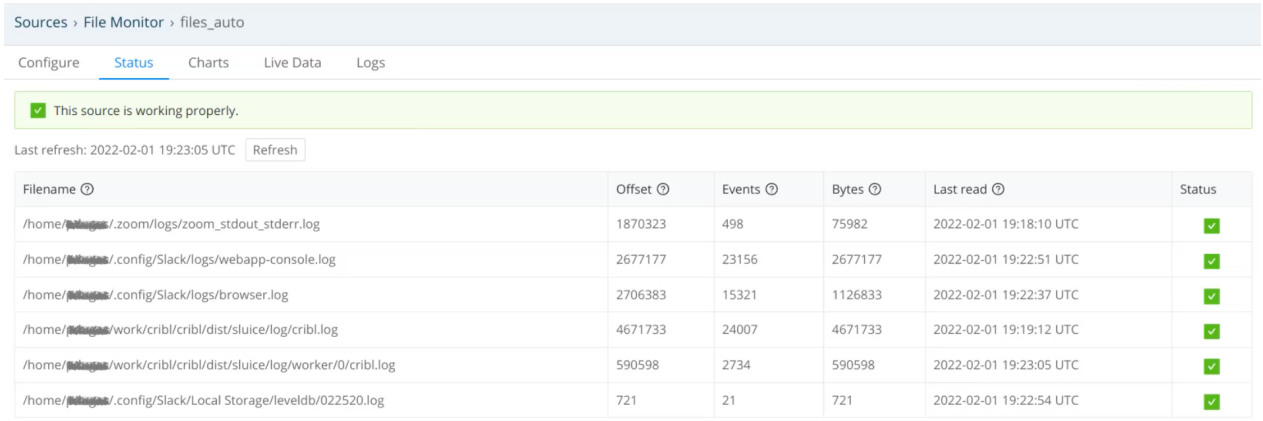
Recall that you'll often create a set of File Monitor Sources, each of which monitors one file. This is **not** the right approach in an example like this one, where **Discovery Mode** is set to **Auto**. Using **Auto** mode in a file monitor along with literally any other file monitor can cause unintended, 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** (e.g., `/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`	`*`

9.1.3. Journal Files

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 and Internal** | TLS Support: **N/A** | Event Breaker Support: **No**
>
> The Journal Files Source is available on Cribl.Cloud hybrid Workers, but not on Cribl-managed Cloud Workers.

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"
> ```

Configuring a Journal Files Source

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Source** at left. From the resulting drawer's tiles, select [**System and Internal** >] **Journal Files**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). From the resulting page's tiles or left nav, select [**System and Internal** >] **Journal Files**. Next, click **Add Source** to open a **New Source** modal that provides the options below.

General Settings

Enabled: Toggle this slider to `Yes` to enable the Source.

Input ID: Enter a unique name to identify this File Monitor Source definition.

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`.

Optional Settings

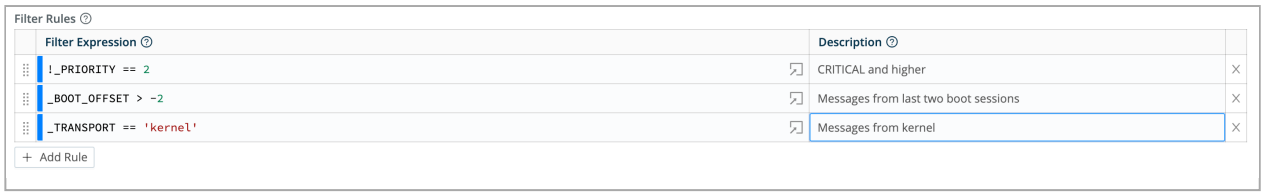
Polling interval: How often, in seconds, to collect metrics. If not specified, defaults to `10` seconds.

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.
- **Description**: Optional description of the rule.

  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

Current boot only: Skip events that are not part of the current boot session.

Max age duration: 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 for filtering and grouping in Cribl Edge. Use a tab or hard return between (arbitrary) tag names.

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.

> In Cribl Edge, this Source defaults to QuickConnect. While, in Cribl Stream, it defaults to Send to Routes.

9.1.4. Kubernetes Events

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 hybrid Workers, but Cribl-managed Workers omit this Source – Cribl manages these Workers' uptime and diagnostics on your behalf.

Discovering and Filtering 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.

> You can bypass the error state by setting the `CRIBL_K8S_FOOTGUN` environment variable to `true`. However, you will assume the risk that the Kubernetes Events Source might make excessive calls to the cluster API.

Configuring Cribl Edge to Collect Kubernetes Events

From the top nav, click **Manage**, then select a **Fleet** to configure. Then, you have two options:

- To configure via the graphical QuickConnect UI, click **Collect**. By default, a **Kubernetes Events** tile appears at left. Hover over it and select **Configure** to open a drawer that provides the options below.
- To configure via the Routing UI, click **More** > **Sources**. From the resulting page's tiles or the **Sources** left nav, select **System and Internal** > **Kubernetes Events**. Next, click the default `in_kube_events` Source to open a modal that provides the options below.

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.

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. 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'`
- **Description**: Optional description of the rule.

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Edge. Use a tab or hard return between (arbitrary) tag names.

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.

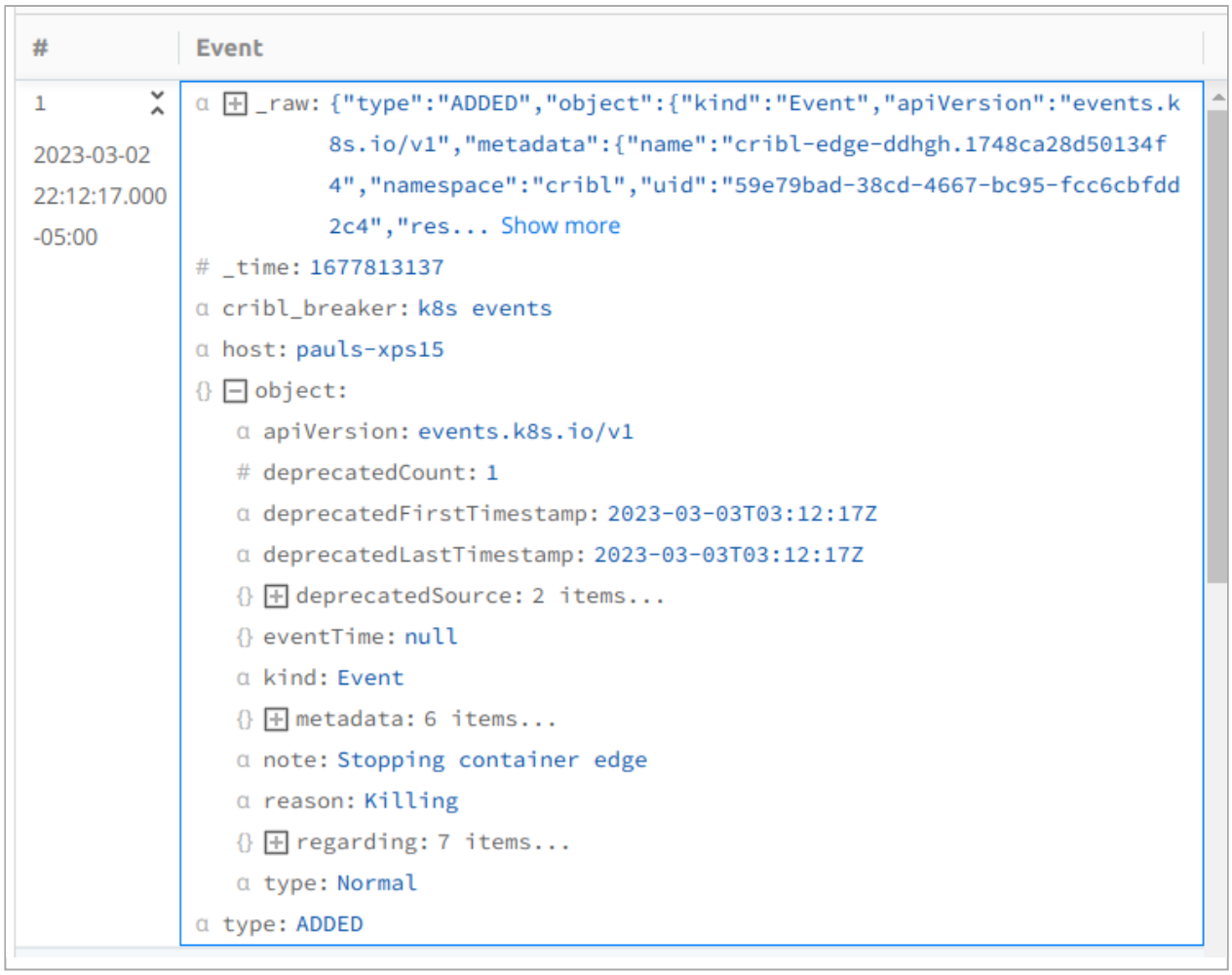
Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

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

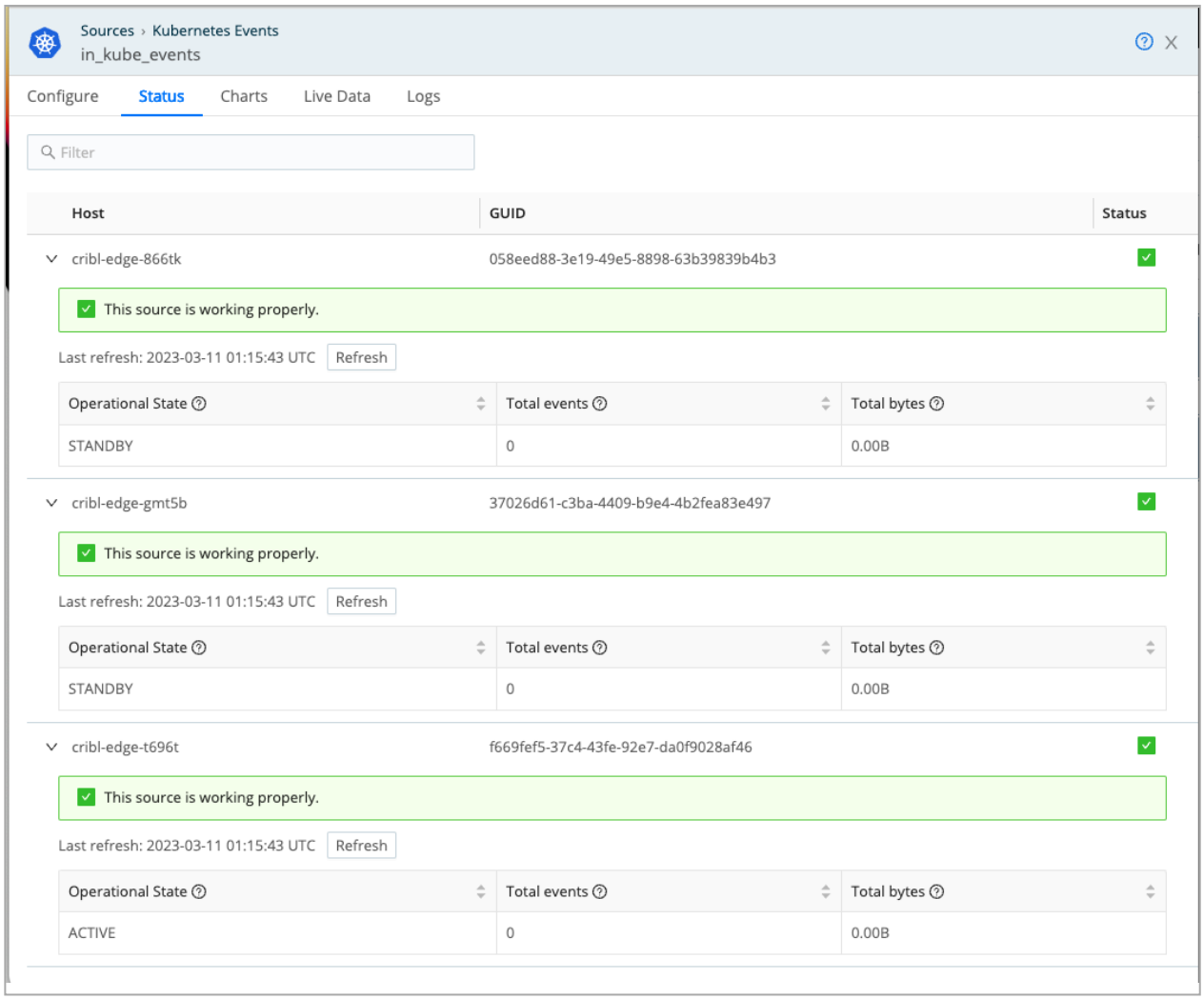
to your Destination(s).



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, click the **Live Data** tab.

Sample Event

9.1.5. Kubernetes Logs

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: **No**
>
> Cribl Edge supports configuring only one Kubernetes Logs Source per Edge Node and per Fleet. This Source is currently unavailable on Cribl-managed Cribl.Cloud Workers.

Discovering and Filtering 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.

> You can bypass the error state by setting the `CRIBL_K8S_FOOTGUN` environment variable to `true`. However, you will assume the risk that the Kubernetes Logs Source might make excessive calls to the cluster API.

Configuring Cribl Edge to Collect Kubernetes Logs

From the top nav, click **Manage**, then select a **Fleet** to configure. Then, you have two options:

- To configure via the graphical QuickConnect UI, click **Collect**. By default, a **Kubernetes Logs** tile appears at left. Hover over it and select **Configure** to open a drawer that provides the options below.
- To configure via the Routing UI, click **More** > **Sources**. From the resulting page's tiles or the **Sources** left nav, select **System and Internal** > **Kubernetes Logs**. Next, click the default `in_kube_logs` Source to open a modal that provides the options below.

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.

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. 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.

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 for filtering and grouping in Cribl Edge. Use a tab or hard return between (arbitrary) tag names.

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**.

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.

9.1.6. Kubernetes Metrics

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 Cribl.Cloud Workers.

Discovering and Filtering 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.

> You can bypass the error state by setting the `CRIBL_K8S_FOOTGUN` environment variable to `true`. However, you will assume the risk that the Kubernetes Metrics Source might make excessive calls to

> the cluster API.

Configuring Cribl Edge to Collect Kubernetes Metrics

From the top nav, click **Manage**, then select a **Fleet** to configure. Then, you have two options:

- To configure via the graphical QuickConnect UI, click **Collect**. By default, a **Kubernetes Metrics** tile appears at left. Hover over it and select **Configure** to open a drawer that provides the options below.
- To configure via the Routing UI, click **More** > **Sources**. From the resulting page's tiles or the **Sources** left nav, select **System and Internal** > **Kubernetes Metrics**. Next, click the default `in_kube_metrics` Source to open a modal that provides the options below.

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.

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. 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'`
- **Description**: Optional description of the rule.

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Edge. Use a tab or hard return between (arbitrary) tag names.

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: What time range of events to hold in each bucket. Default value: `10m` (10 minutes).

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.

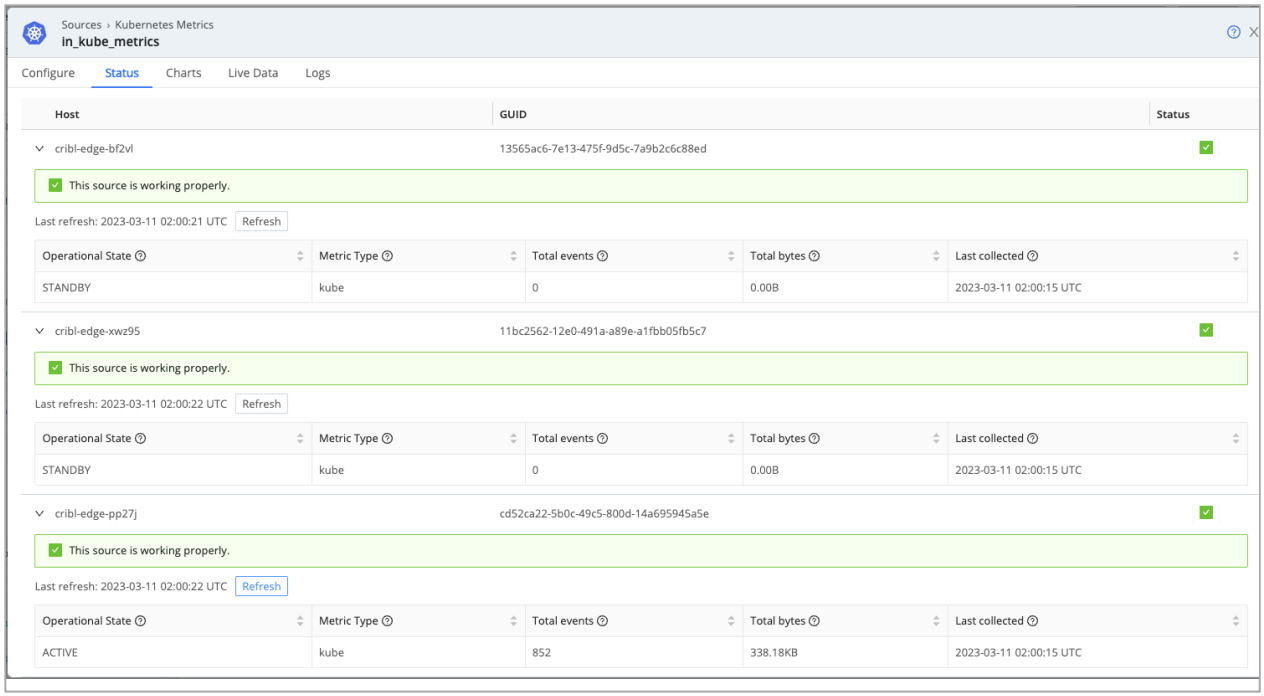
Connected Destinations

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

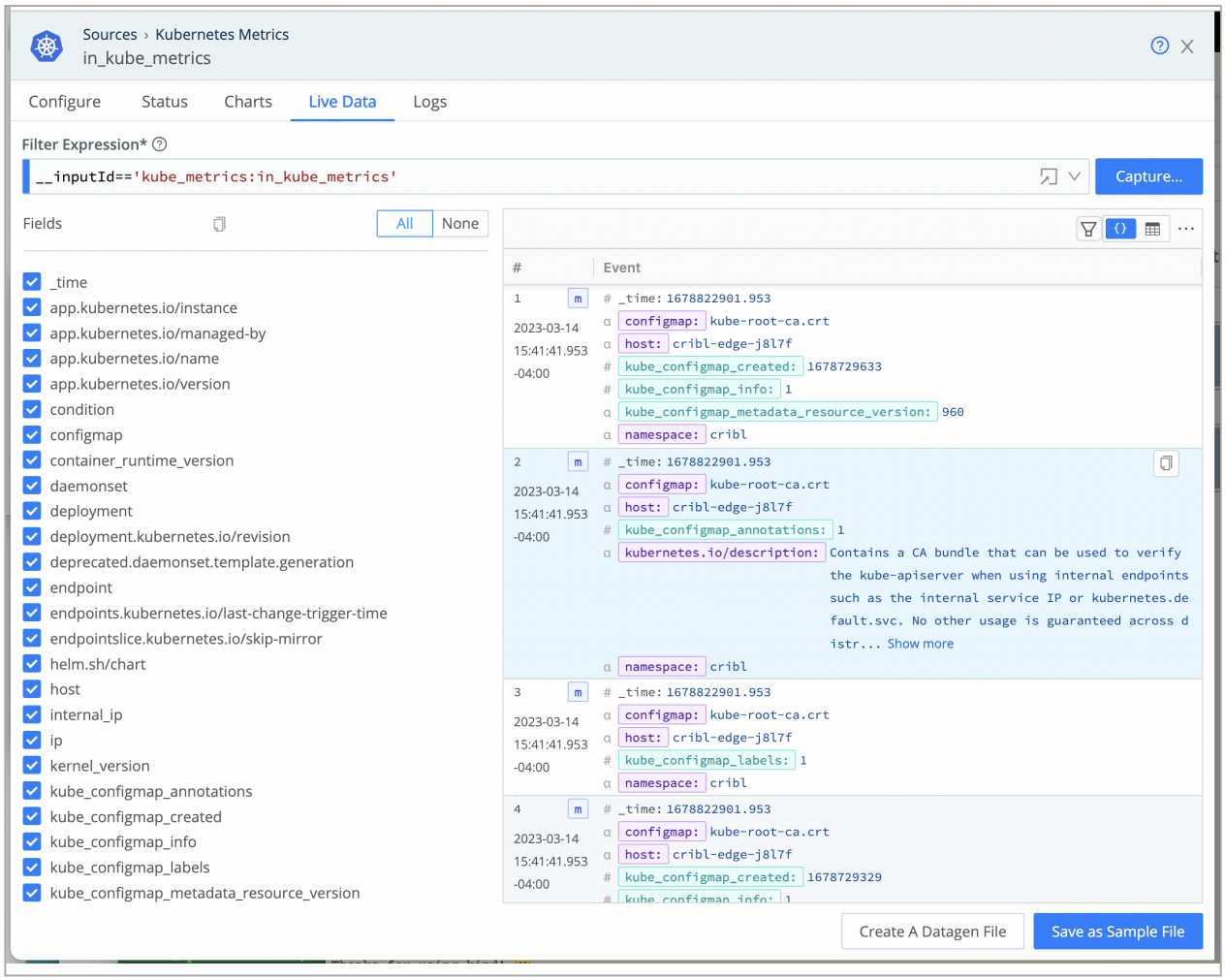
To check the Source's operational state, go to **Status** and expand the host details. The **Operational State** Column shows either an **Active** state or **Standby**. **Active** indicates the Source is running or won the election. **Standby** means it's waiting to be re-elected and not currently running.



Operational State

Source Live Data

To see a sample of the Source's data, click 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

9.1.7. System Metrics

Cribl Edge can collect metrics from the host on which it is running, and can populate some standard metrics dashboards right out of the box. For metrics details, see Linux System Metrics Details.

> Type: **System and Internal** | TLS Support: **N/A** | Event Breaker Support: **No**
>
> Cribl Edge Workers support System Metrics only when running on Linux, not on Windows.
>
> This Source is not availabe for Cribl Stream Workers on Cribl-managed Cribl.Cloud.

Configuring Cribl Edge to Collect System Metrics

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Source** at left. From the resulting drawer's tiles, select [**System and Internal** >] **System Metrics**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). From the resulting page's tiles or left nav, select [**System and Internal** >] **System Metrics**. Next, click **New Source** to open a **New Source** modal that provides the options below.

General Settings

Input ID: Enter a unique name to identify this Source definition.

Optional Settings

Polling interval: How often to collect metrics, in seconds. Defaults to `10`.

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Edge. Use a tab or hard return between (arbitrary) tag names.

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.

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: What time range of events to hold in each bucket. Default value: `10m` (10 minutes).

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.

9.1.8. System State

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 and Internal** | 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 Cribl.Cloud Workers.

Configuring Cribl Edge to Collect System State Events

From the top nav, click **Manage**, then select a **Fleet** to configure. Then, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Source** at left. From the resulting drawer's tiles, select [**System and Internal** >] **System State**. By default, a **System State** tile appears at left. Hover over it and select **Configure** to open a drawer that provides the options below.

Or, to configure via the Routing UI, click **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). From the resulting page's tiles or left nav, select [**System and Internal** >] **System State**. Next, click the default `in_system_state` Source to open modal that provides the options below.

General Settings

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.

Optional Settings

Polling interval: How often, in seconds, to collect system state events. If not specified, defaults to `300s`.

> Each run of the state collector generates events with identical `_time` values.

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Edge. Use a tab or hard return between (arbitrary) tag names.

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** slider 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.

Disks and File Systems

With the **Enabled** slider 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** slider 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.

EVENT FIELD	DESCRIPTION
`search`	Array of search suffixes .
`source`	The source of the data used to populate the event.

Hosts File

With the **Enabled** slider 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** slider 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** slider 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** slider 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** slider 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** slider 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** slider on (the default), Cribl Edge will create events for each configured service (e.g. `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** slider 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: What time range of events to hold in each bucket. Default value: `10m` (10 minutes).

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.

9.2. Internal

9.2.1. Cribl HTTP

The internal Cribl HTTP 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.

> Type: **System and Internal** | TLS Support: **YES** | Event Breaker Support: **No**

You might choose this Source over the Cribl TCP Source in certain circumstances, such as when a firewall or proxy blocks raw TCP egress. In single-instance mode or for testing, you can substitute the Raw HTTP/S Source. (However, this substitution will not provide the single-billing benefits described in the next section.)

How It Works

You can use the Cribl HTTP Source to transfer data between Workers. If the Cribl HTTP Source receives data from its Cribl HTTP Destination counterpart, 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 (on-prem) Edge 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 Edge Nodes 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 Fleet in Cribl.Cloud.
- For an on-prem Edge 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 Cribl.Cloud Edge Nodes, 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 Edge Node that is connected to the same Leader as the Cribl HTTP Source.
- You must specify the same Leader Address on the Edge Nodes that host both the Destination and Source. Otherwise, token verification will fail – breaking the connection and preventing data flow.
- To get the Leader Address specifically for Cribl.Cloud hybrid Workers, see Hybrid Cribl HTTP/Cribl TCP Configuration.
- To configure the Leader Address via the UI, log directly into each Edge Node's UI. Then select **Settings** > **Global Settings** > **Distributed Settings** > **Leader Settings** > **Address**.
- To configure the Leader Address via the instance.yml file, the `host` values on the connecting Edge Nodes must be identical. In this example, both Edge Nodes must point to `cribl-leader`:

```
distributed:
  mode: master
  master:
    host: cribl-leader
    port: 4200
```

- When you configure the Cribl HTTP Destination, its **Cribl endpoint** field must point to the **Address** and **Port** you've configured on the Cribl HTTP Source.
- Finally, it's important to understand the special way the Cribl HTTP Source handles internal fields.

Configuring the Cribl HTTP Source

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Source** at left. From the resulting drawer's tiles, select **System and Internal** > **Cribl HTTP**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). From the resulting page's tile or left nav, select [**System and Internal** >] **Cribl HTTP**. Next, click **New Source** to open a **New Source** modal that provides the options below.

General Settings

Input ID: Enter a unique name to identify this Cribl HTTP Source definition.

Address: Enter the address to bind on. Defaults to `0.0.0.0` (all addresses).

Port: Enter the port number to listen on, e.g., `10200`.

Optional Settings

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Stream. Use a tab or hard return between (arbitrary) tag names.

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 (e.g., 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. E.g., 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

Enable proxy protocol: Toggle to `Yes` if the connection is proxied by a device that supports Proxy Protocol v1 or v2. 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; etc.

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.

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.

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** > **Enable proxy protocol** is set to `No`, the original client IP/port in this header will override the value of `__srcIpPort`.

If **Enable proxy protocol** 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.)

9.2.2. Cribl TCP

The internal Cribl TCP 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.

> Type: **System and Internal** | TLS Support: **YES** | Event Breaker Support: **No**

You might choose this Source over the Cribl HTTP Source in certain circumstances, such as when a firewall or proxy allows raw TCP egress. In single-instance mode or for testing, you can substitute the TCP JSON Source. (However, this substitution will not provide the single-billing benefits described in the next section.)

How It Works

You can use the Cribl TCP Source to transfer data between Workers. If the Cribl TCP Source receives data from its Cribl TCP Destination counterpart, you're billed for ingress only once – when Cribl first receives the data. All data subsequently transferred to other Workers via the Cribl TCP Destination/Source pair is not charged.

This use case is common in hybrid Cribl.Cloud deployments, where a customer-managed (on-prem) Edge 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 Edge to on-prem Stream.

As one usage example, assume that you want to send data from one Edge Nodedeployed 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 Fleet in Cribl.Cloud.
- For an on-prem Edge Nodes 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 Cribl.Cloud Edge Nodes, 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 Edge Node that is connected to the same Leader as the Cribl TCP Source.
- You must specify the same Leader Address on the Edge Nodes that host both the Destination and Source. Otherwise, token verification will fail – breaking the connection and preventing data flow.
- To get the Leader Address specifically for Cribl.Cloud hybrid Workers, see Hybrid Cribl HTTP/Cribl TCP Configuration.
- To configure the Leader Address via the UI, log directly into each Edge Node's UI. Then select **Settings** > **Global Settings** > **Distributed Settings** > **Leader Settings** > **Address**.
- To configure the Leader Address via the instance.yml file, the `host` values on the connecting Edge Nodes must be identical. In this example, both Edge Nodes must point to `cribl-leader`:

```
distributed:
  mode: master
  master:
    host: cribl-leader
    port: 4200
```

- When you configure the Cribl TCP Destination, its **Cribl endpoint** field must point to the **Address** and **Port** you've configured on the Cribl TCP Source.
- Finally, it's important to understand the special way the Cribl TCP Source handles internal fields.

Configuring the Cribl TCP Source

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Source** at left. From the resulting drawer's tiles, select [**System and Internal** >] **Cribl TCP**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). From the resulting page's tiles or left nav, select [**System and Internal** >] **Cribl TCP**. Next, click **New Source** to open a **New Source** modal that provides the options below.

General Settings

Input ID: Enter a unique name to identify this Cribl TCP Source definition.

Address: Enter hostname/IP to listen for TCP JSON data. E.g., `localhost` or `0.0.0.0`.

Port: Enter the port number to listen on, e.g., `10300`.

Additional Settings

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Stream. Use a tab or hard return between (arbitrary) tag names.

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 (e.g., 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. E.g., 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, 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.

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 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` (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
`__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.

9.2.3. Cribl Stream (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: **System and 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 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 a customer-managed (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.

Address: Enter hostname/IP to listen for TCP JSON data. E.g., `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 Cribl Stream. Use a tab or hard return between (arbitrary) tag names.

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 (e.g., 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. E.g., 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

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 Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Enable Persistent Queue** toggle. If enabled, PQ is automatically configured in `Always On` mode, with a maximum queue size of 1 GB disk space allocated per Worker Process.

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 Source. Defaults to `1000`.

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, Cribl Edge will stop queueing data, and will apply the **Queue-full behavior**. Enter a numeral with units of KB, MB, etc. If not specified, the implicit `0` default will enable Cribl Edge to fill all available disk space on the volume.

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

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`

9.2.4. Cribl Internal

The Cribl Internal Source enables you to capture and send Cribl Edge's own internal **logs** and **metrics** through Routes and Pipelines.

> Type: **System and 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.

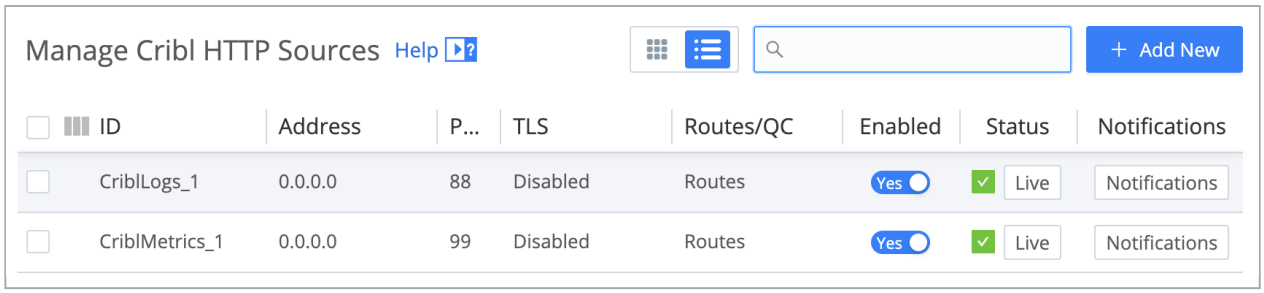
Configuring Cribl Internal Logs/Metrics as a Data Source

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Source** at left. From the resulting drawer's tiles, select [**System and Internal** >] **Cribl Internal**. Next, click **Select Existing**, and click either **Cribl Logs** or **Cribl Metrics** to access the configuration options listed below. When prompted, confirm that you want to switch the existing Source to QuickConnect.

Or, to configure via the Routing UI, click **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). From the resulting page's tiles or left nav, select [**System and Internal** >] **Cribl Internal**. Next, click either **Cribl Logs** or **Cribl Metrics** to open a modal that provides the configuration options listed below.

In either UI, after you've adjusted or confirmed configuration options: On the **CriblLogs** and/or the **CriblMetrics** row, slide the **Enabled** slider to `Yes`. Confirm your choice in the resulting message box. The screenshot below shows both options successfully enabled.



Cribl Internal Sources – click either or both of these rows to configure

CriblLogs – General Settings

Enabled: This duplicates the parent page's **Enabled** slider. Keep it at `Yes` to enable Cribl logs as a Source.

Input ID: Enter a unique name to identify this CriblLogs Source definition.

CriblLogs – Optional Settings

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Edge. Use a tab or hard return between (arbitrary) tag names.

> See also Common Settings below.

CriblMetrics – General Settings

Enabled: This duplicates the parent page's **Enabled** slider. Keep it at `Yes` to enable Cribl metrics as a Source.

Input ID: Enter a unique name to identify this CriblMetrics Source definition.

CriblMetrics – Optional Settings

Metric name prefix: Enter an optional prefix that will be applied to metrics provided by Cribl Edge. The prefix defaults to `cribl.logstream`.

> If Cribl Edge detects `source` or `host` fields in metrics, it copies their values into new dimensions with added `event_` prefixes (e.g., `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** > **System** > **General** > **Limits** > **Metrics** > **Disable field metrics**.

Full fidelity: Toggle this to `No` to exclude granular metrics that can cause high CPU load.

The `No` option will drop the following metrics events:

- `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)`

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Edge. Use a tab or hard return between (arbitrary) tag names.

Processing Settings

Fields

In this section, you can add Fields to each event, using Eval-like functionality. E.g., 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 slider 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.

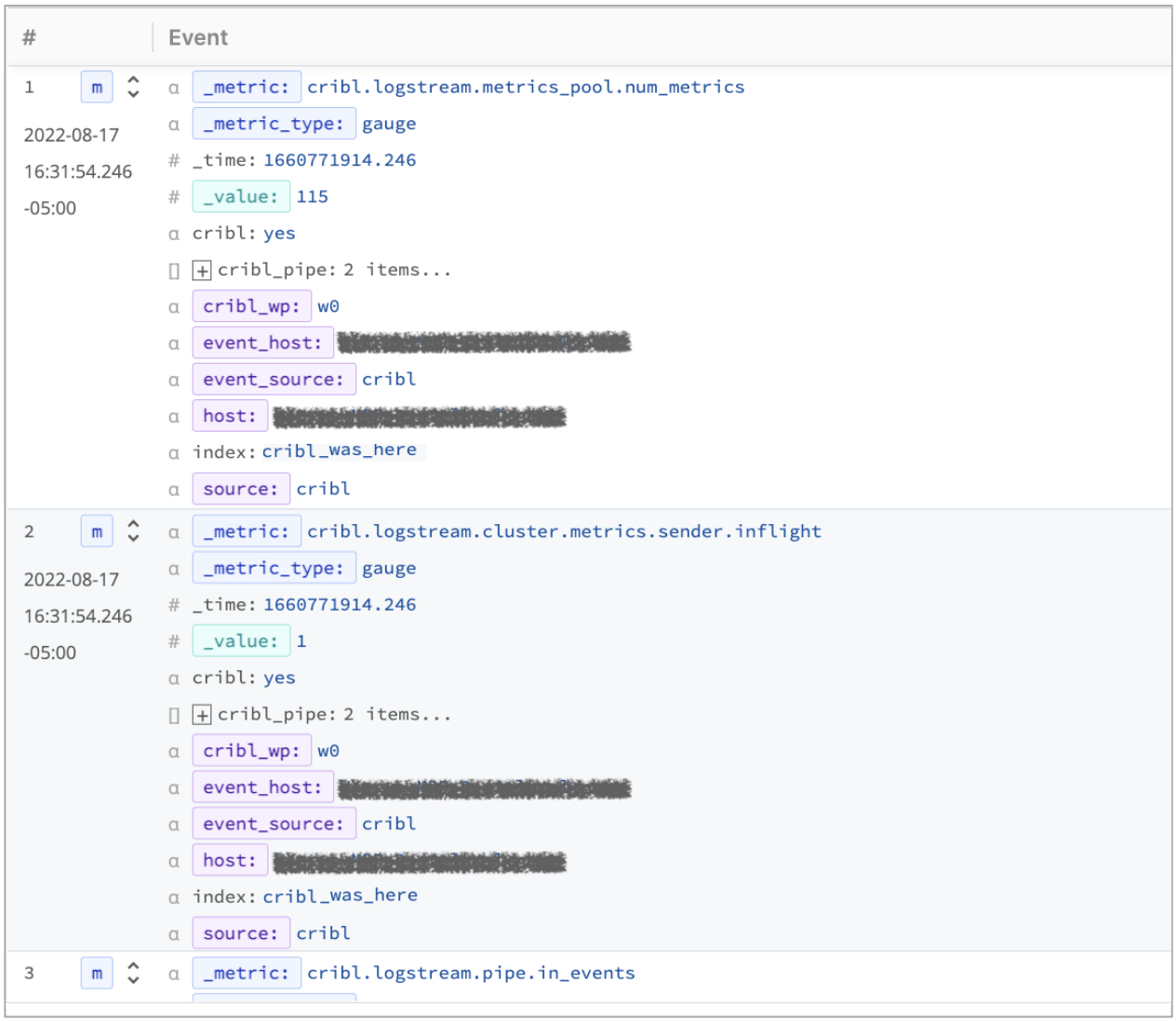
To do this: In the same `cribl_metrics_rollup` pre-processing Pipeline (or a clone) that you attach to your Source, enable the final Eval Function, which applies this **Filter** expression to remove the `sourcetype` field: `_metric && _metric.startsWith('cribl.logstream.sourcetype.')`

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, as covered above in Optional Settings.)

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.

9.2.5. Datagen

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 and Internal** | TLS Support: **N/A** | Event Breaker Support: **No**

Configuring Cribl Edge to Generate Sample Data

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Source** at left. From the resulting drawer's tiles, select [**System and Internal** >] **Datagen**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). From the resulting page's tiles or left nav, select [**System and Internal** >] **Datagen**. Next, click **New Source** to open a **New Source** modal that provides the options below.

General Settings

Input ID: Enter a unique name to identify this Source definition.

Datagens: List of datagens.

- **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`.

Optional Settings

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Edge. Use a tab or hard return between (arbitrary) tag names.

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`

9.3. Amazon

9.3.1. Amazon Kinesis Firehose

Cribl Edge supports receiving data from Amazon Kinesis Data Firehose delivery streams via Kinesis' **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.

Configuring Cribl Edge to Receive Data over HTTP(S) from Amazon Kinesis Firehose

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Source** at left. From the resulting drawer's tiles, select [**Push** >] **Amazon** > **Firehose**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). From the resulting page's tiles or left nav, select [**Push** >] **Amazon** > **Firehose**. Next, click **New Source** to open a **New Source** modal that provides the options below.

General Settings

Input ID: Enter a unique name to identify this Source definition.

Address: Address to bind on. Defaults to 0.0.0.0 (all addresses).

Port: Enter the port number to listen on.

Authentication Settings

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**.

Optional Settings

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Edge. Use a tab or hard return between (arbitrary) tag names.

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 (e.g., 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. E.g., 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

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 Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Enable Persistent Queue** toggle. If enabled, PQ is automatically configured in `Always On` mode, with a maximum queue size of 1 GB disk space allocated per Worker Process.

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 Source. Defaults to `1000`.

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, Cribl Edge will stop queueing data, and will apply the **Queue-full behavior**. Enter a numeral with units of KB, MB, etc. If not specified, the implicit `0` default will enable Cribl Edge to fill all available disk space on the volume.

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

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. 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; etc.

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.

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:

- `__firehoseArn`
- `__firehoseEndpoint`
- `__firehoseReqId`
- `__firehoseToken`
- `__headers` – Added only when **Advanced Settings** > **Capture request headers** is set to `Yes`.
- `__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 Kinesis Firehose client sending data to this Source.

When any proxies (including load balancers) lie between the Kinesis 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** > **Enable proxy protocol** is set to `No`, the original client IP/port in this header will override the value of `__srcIpPort`.

If **Enable proxy protocol** 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/Troubleshooting

If you set the optional `IntervalInSeconds` and/or `SizeInMBs` parameters in the Kinesis 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 Kinesis Firehose.

9.4. Prometheus

9.4.1. Prometheus Edge Scraper

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.

> With v.4.2.x, the deprecated Prometheus Scraper is no longer available on Cribl Edge. Please use this Source instead.

Configuring Cribl Edge to Scrape Prometheus Data

From the top nav, click **Manage**, then select a Fleet to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect**. Next, click **Add Source** at left. From the resulting drawer's tiles, select [**System and Internal** >] **Prometheus Scraper (Edge)**. Next, click either **Add Source** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **More** > **Sources**. From the resulting page's tiles or left nav, select [**System and Internal** >] **Prometheus Edge Scraper**. Next, click **New Source** to open a **New Source** modal that provides the options below.

General Settings

Input ID: Enter a unique name to identify this Source definition.

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`.

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.

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 Cribl Edge. Use a tab or hard return between (arbitrary) tag names.

Authentication (Prometheus)

Use the **Authentication Method** buttons 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 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 Edge Nodes access to authorized AWS resources. Will 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, e.g., 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: What time range of events to hold in each bucket. Default value: `10m` (10 minutes).

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 (e.g., 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`

Proxying Requests

If you need to proxy HTTP/S requests, see System Proxy Configuration.

9.4.2. Prometheus Remote Write

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.

Configuring Cribl Edge to Receive Metrics from Prometheus Remote Write Sources

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Source** at left. From the resulting drawer's tiles, select [**Push** >] **Prometheus** > **Remote Write**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). From the resulting page's tiles or left nav, select [**Push** >] **Prometheus** > **Remote Write**. Next, click **New Source** to open a **New Source** modal that provides the options below.

General Settings

Input ID: Enter a unique name to identify this Source definition.

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`.

Optional Settings

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Edge. Use a tab or hard return between (arbitrary) tag names.

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 click **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 click **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 click **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 (e.g., 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. E.g., 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

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 Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Enable Persistent Queue** toggle. If enabled, PQ is automatically configured in `Always On` mode, with a maximum queue size of 1 GB disk space allocated per Worker Process.

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 Source. Defaults to `1000`.

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, Cribl Edge will stop queueing data, and will apply the **Queue-full behavior**. Enter a numeral with units of KB, MB, etc. If not specified, the implicit `0` default will enable Cribl Edge to fill all available disk space on the volume.

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

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. 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.

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.

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.

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.

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** > **Enable proxy protocol** is set to `No`, the original client IP/port in this header will override the value of `__srcIpPort`.

If **Enable proxy protocol** 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`.

9.4.3. Grafana

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.

Configuring Cribl Edge to Receive Metrics and Logs from Grafana Agent Sources

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Source** at left. From the resulting drawer's tiles, select [**Push** >] **Grafana**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). From the resulting page's tiles or left nav, select [**Push** >] **Grafana**. Next, click **New Source** to open a **New Source** modal that provides the options below.

General Settings

Input ID: Enter a unique name to identify this Source definition.

Address: Enter the hostname/IP to listen to. Defaults to `0.0.0.0`.

Port: Enter the port number to listen on.

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 for filtering and grouping in Cribl Edge. Use a tab or hard return between (arbitrary) tag names.

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 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)

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 (e.g., 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. E.g., 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

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 Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Enable Persistent Queue** toggle. If enabled, PQ is automatically configured in `Always On` mode, with a maximum queue size of 1 GB disk space allocated per Worker Process.

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 Source. Defaults to `1000`.

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, Cribl Edge will stop queueing data, and will apply the **Queue-full behavior**. Enter a numeral with units of KB, MB, etc. If not specified, the implicit `0` default will enable Cribl Edge to fill all available disk space on the volume.

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

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. 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.

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.

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.

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** > **Enable proxy protocol** is set to `No`, the original client IP/port in this header will override the value of `__srcIpPort`.

If **Enable proxy protocol** 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`.

9.4.4. Loki

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.

Configuring Cribl Edge to Receive Loki Logs Data

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Source** at left. From the resulting drawer's tiles, select [**Push** >] **Amazon** > **Loki**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). From the resulting page's tiles or left nav, select [**Push** >] **Amazon** > **Loki**. Next, click **New Source** to open a **New Source** modal that provides the options below.

General Settings

Input ID: Enter a unique name to identify this Source definition.

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`.

Optional Settings

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Edge. Use a tab or hard return between (arbitrary) tag names.

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 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)

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 (e.g., the system's CA).
- **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. E.g., 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

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 Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Enable Persistent Queue** toggle. If enabled, PQ is automatically configured in `Always On` mode, with a maximum queue size of 1 GB disk space allocated per Worker Process.

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 Source. Defaults to `1000`.

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, Cribl Edge will stop queueing data, and will apply the **Queue-full behavior**. Enter a numeral with units of KB, MB, etc. If not specified, the implicit `0` default will enable Cribl Edge to fill all available disk space on the volume.

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

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. 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.

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.

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.

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` – 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 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** > **Enable proxy protocol** is set to `No`, the original client IP/port in this header will override the value of `__srcIpPort`.

If **Enable proxy protocol** 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.)

9.5. Splunk

9.5.1. Splunk HEC

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.

Configuring Cribl Edge to Receive Data over Splunk HEC

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Source** at left. From the resulting drawer's tiles, select [**Push** >] **Splunk** > **HEC**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). From the resulting page's tiles or left nav, select [**Push** >] **Splunk** > **HEC**. Next, click **New Source** to open a **New Source** modal that provides the options below.

> 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.

General Settings

Input ID: Enter a unique name to identify this Splunk HEC Source definition.

Address: Enter the hostname/IP on which to listen for HTTP(S) data. E.g.: `localhost` or `0.0.0.0`. Supports IPv4 and IPv6 addresses.

Port: Enter the port number.

Splunk HEC endpoint: Absolute path on which to listen for the Splunk HTTP Event Collector API requests. Defaults to `/services/collector`.

> This single endpoint supports JSON events via `/event`, raw events via `/raw`, and Splunk S2S events via `/s2s`. See the examples below. The Source will automatically detect where to forward the request.

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`. Some senders might require HEC acks to be enabled. As a result, they might keep TCP connections open while waiting for an ack; this behavior can exhaust available file descriptors. Cribl does not maintain a comprehensive list of senders that require acks – please refer to your sender's documentation.

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Edge. Use a tab or hard return between (arbitrary) tag names.

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 (e.g., 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. E.g., 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

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 Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Enable Persistent Queue** toggle. If enabled, PQ is automatically configured in `Always On` mode, with a maximum queue size of 1 GB disk space allocated per Worker Process.

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 Source. Defaults to `1000`.

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, Cribl Edge will stop queueing data, and will apply the **Queue-full behavior**. Enter a numeral with units of KB, MB, etc. If not specified, the implicit `0` default will enable Cribl Edge to fill all available disk space on the volume.

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

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

If empty (the default), the Splunk HEC Source will permit client access without an auth token. To generate and/or configure tokens, click **Add Token**, which exposes the following fields:

Token: Shared secret to be provided by any client (Authorization: <token>). Click **Generate** to create a new secret. If empty, unauthenticated access **will be permitted**.

Description: Optional description for this token.

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.

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.

Enable proxy protocol: Toggle to `Yes` if the connection is proxied by a device that supports Proxy Protocol v1 or v2. 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; etc.

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.

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`
- `__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** > **Enable proxy protocol** is set to `No`, the original client IP/port in this header will override the value of `__srcIpPort`.

If **Enable proxy protocol** 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 (i.e., 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

```
curl -k http://<myCriblHost>:8088/services/collector/event -H 'Authorization:
myToken42' -d '{"event":"this is a sample event ", "host":"myHost",
"source":"mySource", "fieldA":"valueA", "fieldB":"valueB"}'

curl -k http://<myCriblHost>:8088/services/collector -H 'Authorization: myToken42' -
d '{"event":"this is a sample event ", "host":"myHost", "source":"mySource",
"fieldA":"valueA", "fieldB":"valueB"}'

# Multiple Events
curl -k http://<myCriblHost>:8088/services/collector -H 'Authorization: myToken42' -
d '{"event":"this is a sample event ", "host":"myHost", "source":"mySource",
"fieldA":"valueA", "fieldB":"valueB"}{"event":"this is a sample event 2",
"host":"myHost", "source":"mySource", "fieldA":"valueA", "fieldB":"valueB"}'

# Metrics Events
curl -k http://<myCriblHost>:8088/services/collector/event -H 'Authorization:
myToken42' -d '{"event":"metric", "host":"myHost", "fields":
{"_value":3850,"metric_name":"kernel.entropy_avail"}}'

curl -k http://<myCriblHost>:8088/services/collector/event -H 'Authorization:
myToken42' -d '{"host":"myHost", "fields":
{"_value":3850,"metric_name":"kernel.entropy_avail"}}'

# Send the auth token as a query parameter, with no additional configuration

curl -k "http://<myCriblHost>:8088/services/collector/event?token=mToken42" -d
'{"event":"this is a sample event ", "host":"myHost", "source":"mySource",
"fieldA":"valueA", "fieldB":"valueB"}'
```

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 Endpoint** and port, and the token's value:

```
curl -k "https://in.main-default-<Your-Org-ID>.cribl.cloud:8088/services/collector"
\
    -H "Authorization: <token_value>" \
    -d '{"event": "Goats are better than ponies."}{"event": "Goats are better
climbers."}{"event": "Goats are great yoga buddies.", "nested": {"horns": "Two is
better than one!"}}'
```

9.5.2. Splunk TCP

Cribl Edge supports receiving Splunk data from Universal or Heavy Forwarders.

> Type: **Push** | TLS Support: **YES** | Event Breaker Support: **YES**

Configuring Cribl Edge to Receive Splunk TCP Data

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Source** at left. From the resulting drawer's tiles, select [**Push** >] **Splunk** > **Splunk TCP**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). From the resulting page's tiles or left nav, select [**Push** >] **Splunk** > **Splunk TCP**. Next, click **New Source** to open a **New Source** modal that provides the options below.

> 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.

General Settings

Input ID: Enter a unique name to identify this Splunk Source definition.

Address: Enter hostname/IP to listen for Splunk data. E.g., `localhost` or `0.0.0.0`.

Port: Enter port number.

Optional Settings

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Edge. Use a tab or hard return between (arbitrary) tag names.

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 (e.g., the system's CA). Defaults to `Yes`.
- **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. E.g., 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

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 Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Enable Persistent Queue** toggle. If enabled, PQ is automatically configured in `Always On` mode, with a maximum queue size of 1 GB disk space allocated per Worker Process.

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 Source. Defaults to `1000`.

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, Cribl Edge will stop queueing data, and will apply the **Queue-full behavior**. Enter a numeral with units of KB, MB, etc. If not specified, the implicit `0` default will enable Cribl Edge to fill all available disk space on the volume.

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 of 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 `.*` (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.

Max S2S version: The highest version of the Splunk-to-Splunk protocol to expose during handshake. Defaults to `v3`; `v4` 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.

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.

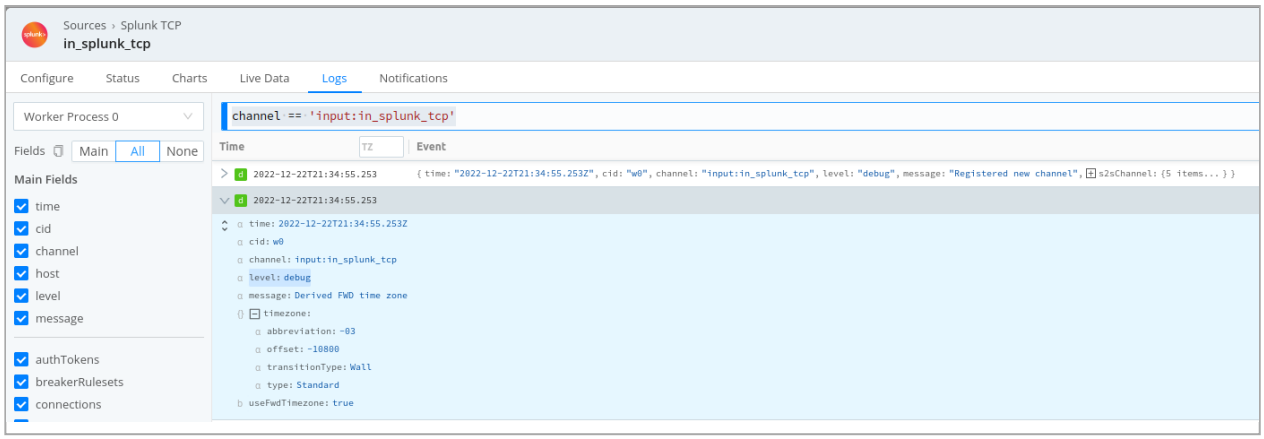
Using the UF Time Zone

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
172.17.0.1:9997 failed. Connection refused
```

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

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`
- `__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>,

[tcpout:cribl]
server = [<cribl_ip>|<cribl_host>]:<port>, [<cribl_ip>|<cribl_host>]:<port>, ...
sendCookedData = true
# As of Splunk 6.5, using forceTimebasedAutoLB is no longer recommended. Ensure this
is left at default for UFs
# forceTimebasedAutoLB = false
negotiateProtocolLevel = 0
```

Preventing Data Loss with v3

If you set **Max S2S version** to `v3`, Cribl recommends that you use the `negotiateProtocolLevel = 0` setting shown above to avoid data loss. Depending on your environment, enabling `negotiateProtocolLevel` with a non-`0` value could cause Cribl to not accept data from the forwarder.

If you set **Max S2S version** to `v4`, this setting is 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 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.

9.6. Windows

9.6.1. Windows Event Forwarder

Cribl Edge supports receiving Windows events from the Windows Event Forwarding mechanism built into modern versions of Microsoft Windows (including Windows 10, Windows Server 2012, and more-recent releases).

> Type: **Push** | TLS Support: **YES** | Event Breaker Support: **No**

This Source supports client certificates (mutual TLS) authentication and Kerberos authentication on Linux.

On Cribl Edge, this Source currently supports mutual TLS only with Windows Edge Nodes.

Upstream Prerequisites

This page assumes that you:

- Already have Windows Event Forwarding set up.
- Are pointed to one or more Windows Event Collectors (WECs).
- Are using either client certificate authentication over HTTPS or Kerberos authentication over HTTP.

If you are using client certificate authentication, we also assume that you have – or will generate – a server certificate for this Source, issued by the same Certificate Authority as the client certs.

> For details, see Configuring Upstream Clients/Senders.
>
> For a complete walk-through of generating certificates, setting permissions and policies, and ingesting Windows Event subscriptions and data through Cribl.Cloud, see our Configuring WEF for Cribl Stream topic.

Configuring Cribl Edge to Receive Windows Events

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Source** at left. From the resulting drawer's tiles, select [**Push** >] **Windows Event Forwarder**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). From the resulting page's tiles or left nav, select [**Push** >] **Windows Event Forwarder**. Next, click **New Source** to open a **New Source** modal that provides the options below.

General Settings

Input ID: Enter a unique name to identify this Windows Event Forwarder Source definition.

Address: Enter the hostname/IP on which to listen for Windows events data. (E.g., `localhost` or `0.0.0.0`.)

Port: Enter the port number. The default, 5986, is the port used by Windows Event Collector for HTTPS-based subscriptions.

Authentication method: Specify the method for accepting incoming client connections – either `Client certificate` or `Kerebros`.

Client Certification Authentication Settings

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`. If multiple certificates are present in a `.pem`, each must directly certify the one preceding it in that file. 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 be the same 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 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. E.g., 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.

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 this additional 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).

Kerberos Authentication Settings

To implement Kerberos authentication, you must first prepare your environment.

Service principal name: Specify the full service principal name, in the form `HTTP/<fully qualified domain name>@REALM`. This is case-sensitive.

Keytab location: Specify the path to the keytab file containing the service principal credentials. Cribl Edge will use `/etc/krb5.keytab` if you do not specify a different keytab file.

> Kerberos authentication is disabled for Cribl-managed Cribl.Cloud Workers, but it is enabled for on-prem Worker Groups, whether the Leader is based in Cribl.Cloud or on-prem.

Optional Settings

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Edge. Use a tab or hard return between (arbitrary) tag names.

Subscriptions

Click **Add Subscription** to define a new subscription. At least one subscription is required, and has the following options:

Name: A friendly name for the subscription, which is only used to help you identify it.

Version: A read-only field which will be populated when the Source is saved, and will be assigned a new value any time new changes are made to the subscription that affect how a client interprets the subscription.

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 aggregate 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.

Read existing events: Control whether historical events should be sent by the client when it first connects. If set to `No`, only events generated by the client after the subscription is received will be forwarded. If set to `Yes`, the behavior depends on the **Use bookmarks** setting:

- If **Use bookmarks** is set to `No`, then each time a client *forcibly* renews its subscription (e.g., after group policy update), all events matching the query will be sent. Restarting Cribl Edge or restarting the client will not cause all events to be re-sent.
- If **Use bookmarks** is set to `Yes`, then when the client receives a subscription for the first time, it will send all historical events matching the query, but on subsequent (even forced) resubscriptions, it will only send events later than the saved bookmark.
- In either case, if a subscription is changed and saved, 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.

Use bookmarks: If toggled to `Yes`, Cribl Edge will keep track of which events have been received, resuming from that point even if the client forcibly re-subscribes. If set to `No`, a client (re-)subscribing will either send all historical events, or send only events subsequent to the subscription, depending on the **Read existing events** setting.

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. Wildcard-matching is supported.

Configuring 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. You have two **Query builder mode** options: **Simple** or **Raw XML**.

Select **Raw XML** if you have an existing XPath query. Paste your query into the **XML query** field.

Select **Simple** if you need 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 enter the DNS names of the endpoints that should forward these events. Supports wildcards (e.g., `*.mydomain.com`). The default global wildcard (`*`) enables all endpoints.

> If you have both **Simple** and **Raw XML** queries defined within a given Subscription, the **Query builder mode** button that you select when saving this Source's configuration determines which query/queries Cribl Edge will send. To send both **Simple** and **Raw XML** queries, use the **Add Subscription** button to define multiple subscriptions.

Example

When creating a subscription in a Windows Event Collector, you can view the generated XML/XPath query. Consider a subscription that returns all events from the Security log that are of severities Critical, Error, or Warning, and that occurred within the last 24 hours. This subscription would generate XML like this:

```
<QueryList>
  <Query Id="0" Path="Security">
    <Select Path="Security">*[System[(Level=1  or Level=2 or Level=3) and
TimeCreated[timediff(@SystemTime) &lt;= 86400000]]]</Select>
  </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 Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Enable Persistent Queue** toggle. If enabled, PQ is automatically configured in `Always On` mode, with a maximum queue size of 1 GB disk space allocated per Worker Process.

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 Source. Defaults to `1000`.

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, Cribl Edge will stop queueing data, and will apply the **Queue-full behavior**. Enter a numeral with units of KB, MB, etc. If not specified, the implicit `0` default will enable Cribl Edge to fill all available disk space on the volume.

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

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.

Enable proxy protocol: Toggle to `Yes` if the connection is proxied by a device that supports Proxy Protocol v1 or v2. 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.

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.

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 `5` 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.

CA fingerprint override: Use this setting only if the first certificate in the configured CA chain does not match the SHA1 fingerprint that the WEF client expects. If that is the case, enter the expected SHA1 fingerprint.

This setting only applies to client certificate authentication. It is ignored if you use Kerberos authentication.

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 authentation so it can avoid the overhead of performing the two-step Kerberos authentication on each new batch delivery.

If the socket timeout has elapsed between event deliveries, Cribl Edge will still handle it gracefully, but a small additional overhead is incurred.

Setting **Keep-alive timeout** >= **Batch timeout** will prevent this additional re-authentication step, but it also means that Cribl Edge will maintain open socket connections for each client for longer, which could cause it to hit the limit set by **Advanced Settings** > **Max active requests**. As a result, Cribl Edge will no longer accept additional connections for a given worker process.

Keeping the **Keep-alive timeout** low (such as the default 5 seconds) will minimize Cribl Edge memory usage (due to long-lived open connections), but it will result in an additional network round-trip on each batch of events delivered.

Note that the **Subscriptions** > **Heartbeat** setting is unaffected by the above because Windows treats socket connections for heartbeat signals as one-time-use and immediately closes them.

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`
- `__srcIpPort` – See details below.
- `__subscriptionName`
- `__subscriptionVersion`
- `__headers` – Added only when **Advanced Settings** > **Capture request headers** is set to `Yes`.

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.

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** > **Enable proxy protocol** is set to `No`, the original client IP/port in this header will override the value of `__srcIpPort`.

If **Enable proxy protocol** 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 Upstream Clients/Senders

This section summarizes how to configure Windows endpoints/senders to forward events to this Cribl Edge Source. You can find basic instructions for setting up WEF (in a traditional Windows environment) in this Microsoft guide. You can generally follow that guide's "non-domain" section to correctly configure the endpoints/senders.

> For a complete walk-through of generating certificates, setting permissions and policies, and ingesting Windows Event subscriptions and data through Cribl.Cloud, see Cribl's Configuring WEF for Cribl Stream topic.

1. Set up Cribl Edge's Windows Event Forwarder Source as outlined above. The CA certificate you use should be the issuing authority both for the server certificate, and for all client certs you plan to have forwarding events to this Source.

2. Ensure that your existing Windows Event Collector is receiving events correctly from whatever endpoints and subscriptions you already have in place.

3. For client certificate authentication only: Find the fingerprint of the CA cert you are using for this Source, via a tool like `certutil` or `certlm` on Windows, or `openssl` on other operating systems. You'll need this in the next step.

4. Edit the group policy for endpoints you want to forward to this Source:

    - Under **Computer Configuration > Administrative Templates > Windows Components > Event Forwarding**, modify the **Configure target Subscription Manager** setting.

    - Add a Subscription Manager with a value like: `Server=https://<cribl-instance>:<wef-source-port>/wsman/SubscriptionManager/WEC,Refresh=<desired refresh interval>,IssuerCA=<CA cert fingerprint from above>`

        - For Kerberos authentication, use `http://`, not `https://`. Kerberos provides its own encryption and does not use TLS.
        - For Kerberos authentication, do not enter the `IssuerCA` portion.

    - Note that the path portion is the same as required for a WEC subscription, and is not configurable in Cribl Edge.

- Ensure that the protocol is one of the following:
  - `https` for client certificate (mutual TLS) authentication.
  - `http` (for Kerberos authentication).
- When complete, save the policy and apply it to affected endpoints.

5. Check that events are flowing into Cribl Edge now according to the configured subscriptions. If they are not:
   - Certificate authentication only: Verify that 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 Edge via a proxy, you may need to set **Enable proxy protocol** to `Yes` (in the **Advanced** section of the Source configuration). Ensure that the correct outbound firewall port is opened on the client.
   - Certificate authentication only: Verify all of the following: that the certificate chain is correct; that the endpoints have a valid CRL encompassing the CA cert; that the CA cert is a trusted root on the clients; and that the server and client certs are issued by the same CA. The `CAPI2` Windows event log might reveal any errors here.
   - Certificate authentication or Kerberos authentication: Check Cribl Edge for any errors, as well as the `EventForwarding-Plugin` and `Windows Remote Management` event logs on the clients.

Preparing the Environment for Kerberos Authentication

This section documents the setup steps you need to take before configuring a Windows Event Forwarder Source for Kerberos authentication.

Initial Conditions and Assumptions

We assume that:

- 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`.

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
```

Setup for the Domain Controller

To set up the domain controller:

1. Set up forward and reverse DNS for the Edge Node's FQDN (i.e., 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`.

Keytab Export

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 `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`.

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.

> 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`.

Setup for the Windows Event Forwarder Source

The general setup for creating subscriptions is the same as documented in Subscriptions.

When you create the Source, select the `Kerberos` authentication option. Specify the SPN and keytab file location that you set above in Keytab Export. The SPN is case-sensitive, and must be exactly in the form used to export the keytab.

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

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

1. If you haven't already done so, create a GPO to auto-start the WinRM service on the client machines:

2. Access the UI **Computer Configuration** > **Policies** > **Windows Settings** > **Security Settings** > **System Services** > **Windows Remote Management (WS-Management)**.

3. Set **Startup type** to `Automatic`.

Create a GPO for the NETWORK SERVICE

1. If you haven't already done so, create a GPO so that the NETWORK SERVICE can read event logs (required if you want to subscribe to the Security event log):

2. Access the UI **Computer Configuration** > **Policies** > **Windows Settings** > **Security Settings** > **Restricted Groups**.

3. In the right pane, right-click and select **Add Group**.

4. Enter `Event Log Readers` as the group name.

5. Under `Members of this group`, click **Add**.

6. Enter `NETWORK SERVICE` as the member name.

> For the change to take effect, you'll have to restart computers after you apply this GPO.

When you have finished up your environment, select the appropriate settings for Kerberos authentication.

Troubleshooting

When using auto-enroll, the `Eventlog-ForwardingPlugin Operational` logs might display an error of this form:

```
If Kerberos mechanism is used, verify that the client computer and the destination
computer are joined to a domain.
```

To resolve this error: In your auto-enroll template's **Client-Server Authentication Properties**, make sure the **Subject Name format** is set to **Common name**.

9.6.2. Windows Event Logs

Cribl Edge supports collecting local Windows Event Logs in batches. You can collect the standard event logs – Application, Security, and System – and any other event logs on the machine.

> Type: **Pull** | TLS Support: **N/A** | Event Breaker Support: **No**
>
> Cribl Edge Workers support Windows Event Logs only when running on Windows, not on Linux.

Configuring Cribl Edge to Collect Windows Event Logs

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

Configure via QuickConnect

1. To configure via the graphical QuickConnect UI, click **Collect** (Edge only).
2. Click **Add Source** at left. From the resulting drawer's tiles, select **Push** > **Windows Event Logs**.
3. Click either **Add Destination** or (if displayed) **Select Existing**.

The **General Settings** drawer will open.

Configure via Routing

1. Click **Data** > **Sources** (Stream) or **More** > **Sources** (Edge).
2. From the resulting page's tiles or left nav, select **Push** > **Windows Event Logs**.
3. Click **Add Source** to open the **New Source** modal.

General Settings

Input ID: Enter a unique name to identify this Source definition.

Log Name: Enter one or more event logs to collect. `Security` is prefilled as a default. Click **Add Name** to specify more event logs (e.g., `Application` or `System`).

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.

- It may be faster to render events as XML.
- JSON format includes the rendered message string, while XML does not.

  Sample JSON


```
{
"_raw": "

{\"Id\":1234,\"Version\":0,\"Qualifiers\":null,\"Level\":0,\"Task\":9999,\"Opcode\"
Windows-Security-Auditing\",\"ProviderId\":\"GUID-12345678-1234-1234-1234-
123456789012\",\"LogName\":\"Security\",\"ProcessId\":123,\"ThreadId\":5678,\"Machi
HOST\",\"UserId\":null,\"TimeCreated\":\"\\/Date(1000000000000)\\/\",\"ActivityId\"
123456789012\",\"RelatedActivityId\":null,\"ContainerLog\":\"Security\",\"MatchedQu
{},\"LevelDisplayName\":\"Information\",\"OpcodeDisplayName\":\"Info\",\"TaskDispla
Success\"],\"Properties\":[{\"Value\":\"Administrator\"},{\"Value\":\"EC2-HOST\"},{
{\"Value\":\"Administrator\"},{\"Value\":\"EC2-HOST\"},{\"Value\":9999999},{\"Value
message about user and process information.\"}",
"source": "Security",
"host": "EC2-HOST",
"_time": "1000000000.000",
"cribl_breaker": "windows event logs (json)"
  }
```


  Sample XML

```
   {
"_raw": "<Event xmlns='http://schemas.microsoft.com/win/2004/08/events/event'>
<System><Provider Name='Microsoft-Windows-Security-Auditing' Guid='{GUID-
12345678-1234-1234-1234-123456789012}'/><EventID>1234</EventID>
<Version>0</Version><Level>0</Level><Task>12345</Task><Opcode>0</Opcode>
<Keywords>0x8020000000000000</Keywords><TimeCreated SystemTime='2000-01-
01T00:00:00.0000000Z'/><EventRecordID>123456</EventRecordID><Correlation
ActivityID='{12345678-1234-1234-1234-123456789012}'/><Execution ProcessID='1234'
ThreadID='5678'/><Channel>Security</Channel><Computer>GENERIC-HOST</Computer>
<Security/></System><EventData><Data Name='TargetUserName'>GenericUser</Data>
<Data Name='TargetDomainName'>GenericDomain</Data><Data Name='TargetSid'>SID-
12345678</Data><Data Name='SubjectUserSid'>SID-87654321</Data><Data
Name='SubjectUserName'>Administrator</Data><Data Name='SubjectDomainName'>EC2-
Host</Data><Data Name='SubjectLogonId'>0x123456</Data><Data
Name='CallerProcessId'>0x7890</Data><Data Name='CallerProcessName'>C:\\Program
Files\\PowerShell\\7\\pwsh.exe</Data></EventData></Event>",
"source": "Security",
"host": “EC2-Host,
"_time": "1000000000.000",
"cribl_breaker": "windows event logs (xml)"
  }
```


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.

Tags: Add tags to use for filtering and grouping in Cribl Edge. Use a tab, space, or hard return between (arbitrary) tag names.

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 Event Bytes: The maximum number of bytes that can be in an event before it is flushed 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

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.

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`.

9.6.3. Windows Metrics

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. Edge Leaders support configuring only one Windows Metrics Source per Fleet.

Configuring Cribl Edge/Windows to Collect System Metrics

From the top nav, click **Manage**, then select a **Fleet** to configure. Then, you have two options:

- To configure via the graphical QuickConnect UI, click **Collect**. By default, a **Windows Metrics** tile appears at left. Hover over it and select **Configure** to open a drawer that provides the options below.
- To configure via the Routing UI, click **More** > **Sources**. From the resulting page's tiles or the **Sources** left nav, select **System and Internal** > **Windows Metrics**. Next, click the default `in_windows_metrics` Source to open a modal that provides the options below.

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.

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 Cribl Edge. Use a tab or hard return between (arbitrary) tag names.

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.

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: What time range of events to hold in each bucket. Default value: `10m` (10 minutes).

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

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.

9.7. AppScope

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.

> Type: **Push** | TLS Support: **YES** | Event Breaker Support: **YES**

Configuring Cribl Edge to Receive AppScope Data

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Source** at left. From the resulting drawer's tiles, select [**System and Internal** >] **AppScope**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). From the resulting page's tiles or left nav, select [**System and Internal** >] **AppScope**. Next, click **New Source** to open a **New Source** modal that provides the options below.

General Settings

Input ID: Enter a unique name to identify this AppScope Source definition.

UNIX domain socket: When toggled to `Yes`, 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, e.g., `777`. If empty, Cribl Edge will use the runtime user's default permissions.

When **UNIX domain socket** is set to `No` (the default), 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. (E.g., `localhost`.) Defaults to `0.0.0.0`, meaning all addresses.
- **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 clients must provide in the `authToken` header field. Click **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.

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 for filtering and grouping in Cribl Edge. Use a tab or hard return between (arbitrary) tag names.

TLS Settings (Server Side)

This left tab is displayed only when the Optional Settings > **UNIX domain socket** slider 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 (e.g., the system's CA). Defaults to `Yes`.

- **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. E.g., 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.

AppScope Rules

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 Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Enable Persistent Queue** toggle. If enabled, PQ is automatically configured in `Always On` mode, with a maximum queue size of 1 GB disk space allocated per Worker Process.

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 Source. Defaults to `1000`.

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, Cribl Edge will stop queueing data, and will apply the **Queue-full behavior**. Enter a numeral with units of KB, MB, etc. If not specified, the implicit `0` default will enable Cribl Edge to fill all available disk space on the volume.

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.

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: What time range of events to hold in each bucket. Default value: `10m` (10 minutes).

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
```

9.8. Datadog Agent

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`.

Configuring Cribl Edge to Ingest Datadog Agent Output

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Source** at left. From the resulting drawer's tiles, select [**Push** >] **Datadog Agent**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). From the resulting page's tiles or left nav, select [**Push** >] **Datadog Agent**. Next, click **New Source** to open a **New Source** modal that provides the options below.

General Settings

Input ID: Enter a unique name to identify this Datadog Agent Source definition.

Address: Enter hostname/IP to listen for Datadog Agent data. E.g., `localhost` or `0.0.0.0`.

Port: Enter the port number to listen on.

Optional Settings

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Edge. Use a tab or hard return between (arbitrary) tag names.

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 (e.g., the system's CA). Defaults to `Yes`.
- **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. E.g., 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

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 Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Enable Persistent Queue** toggle. If enabled, PQ is automatically configured in `Always On` mode, with a maximum queue size of 1 GB disk space allocated per Worker Process.

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 Source. Defaults to `1000`.

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, Cribl Edge will stop queueing data, and will apply the **Queue-full behavior**. Enter a numeral with units of KB, MB, etc. If not specified, the implicit `0` default will enable Cribl Edge to fill all available disk space on the volume.

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

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. 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 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.

- **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 (e.g., the system's CA). Defaults to `No`. Available when **Forward API key validation requests** is toggled to `Yes`.

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.

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** > **Enable proxy protocol** is set to `No`, the original client IP/port in this header will override the value of `__srcIpPort`.

If **Enable proxy protocol** 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, e.g., `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.

9.9. Elasticsearch API

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.

Configuring Cribl Edge to Receive Elasticsearch Bulk API Data over HTTP(S)

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Source** at left. From the resulting drawer's tiles, select [**Push** >] **Elasticsearch API**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). From the resulting page's tiles or left nav, select [**Push** >] **Elasticsearch API**. Next, click **New Source** to open a **New Source** modal that provides the options below.

> 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.

General Settings

Input ID: Enter a unique name to identify this Elasticsearch Source definition.

Address: Enter the hostname/IP on which to listen for Elasticsearch data. (E.g., `localhost` or `0.0.0.0`.)

Port: Enter the port number.

Elasticsearch API endpoint (for Bulk API): Absolute path on which to listen for Elasticsearch API requests. Defaults to `/`. Cribl Edge automatically appends `_bulk`, so (e.g.) `/myPath` becomes `/myPath/_bulk`. Requests could then be made to either `/myPath/_bulk` or `/myPath/<myIndexName>/_bulk`. Other entries are faked as success.

Optional Settings

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Edge. Use a tab or hard return between (arbitrary) tag names.

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 click **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.

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 (e.g., the system's CA). Defaults to `Yes`.
- **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. E.g., 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

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 Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Enable Persistent Queue** toggle. If enabled, PQ is automatically configured in `Always On` mode, with a maximum queue size of 1 GB disk space allocated per Worker Process.

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 Source. Defaults to `1000`.

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, Cribl Edge will stop queueing data, and will apply the **Queue-full behavior**. Enter a numeral with units of KB, MB, etc. If not specified, the implicit `0` default will enable Cribl Edge to fill all available disk space on the volume.

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

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. 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; etc.

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.

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.)

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, e.g.: `http://elastic:9200`.

Reject unauthorized certificates: Reject certificates that are not authorized by a trusted CA (e.g., the system's CA). Defaults to `No`.

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.

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 Settings

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`
- `__srcIpPort` – See details below.
- `__type`

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** > **Enable proxy protocol** is set to `No`, the original client IP/port in this header will override the value of `__srcIpPort`.

If **Enable proxy protocol** 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.

9.10. HTTP/S (Bulk API)

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.

Configuring Cribl Edge to Receive Data over HTTP(S)

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Source** at left. From the resulting drawer's tiles, select [**Push** >] **HTTP**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). From the resulting page's tiles or left nav, select [**Push** >] **HTTP**. Next, click **New Source** to open a **New Source** modal that provides the options below.

> 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.

General Settings

Input ID: Enter a unique name to identify this HTTP(S) Source definition.

Address: Enter the hostname/IP on which to listen for HTTP(S) data. (E.g., `localhost` or `0.0.0.0`.)

Port: Enter the port number.

Authentication Settings

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**.

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. Currently, the only supported option is the default `/elastic`, to which Cribl Edge will append `/_bulk`. So, your senders should send events to an `/elastic/_bulk` path. Other entries are faked as success. Use an empty string to disable.

> Cribl generally recommends that you use the dedicated Elasticsearch API Source instead of this endpoint. The Elastic API implementation here is provided for backward compatibility, and for users who want to ingest multiple inputs on one HTTP/S port.

Splunk HEC endpoint: 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`.

> This Splunk HEC implementation is an **event** (i.e., 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.

Splunk HEC Acks: Whether to enable Splunk HEC acknowledgements. Defaults to `No`.

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Edge. Use a tab or hard return between (arbitrary) tag names.

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 (e.g., the system's CA). Defaults to `Yes`.
- **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. E.g., 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

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 Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Enable Persistent Queue** toggle. If enabled, PQ is automatically configured in `Always On` mode, with a maximum queue size of 1 GB disk space allocated per Worker Process.

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 Source. Defaults to `1000`.

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, Cribl Edge will stop queueing data, and will apply the **Queue-full behavior**. Enter a numeral with units of KB, MB, etc. If not specified, the implicit `0` default will enable Cribl Edge to fill all available disk space on the volume.

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

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. 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; etc.

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.

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** > **Enable proxy protocol** is set to `No`, the original client IP/port in this header will override the value of `__srcIpPort`.

If **Enable proxy protocol** 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":"mySource", "fieldA":"valueA", "fieldB":"valueB"}
{"_time":1541280341, "host":"myOtherHost", "source":"myOtherSource", "_raw": "
{\"message\":\"Something informative happened\", \"severity\":\"INFO\"}"}
```

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
'{"_raw":"this is a sample event ", "host":"myHost", "source":"mySource",
"fieldA":"valueA", "fieldB":"valueB"}'
```

Cribl Endpoint – Multiple Events

```
curl -k http://<myCriblHost>:10080/cribl/_bulk -H 'Authorization: myToken42' -d
$'{"_raw":"this is a sample event ", "host":"myHost", "source":"mySource",
"fieldA":"valueA", "fieldB":"valueB"} \n {"_raw":"this is another sample event ",
"host":"myOtherHost", "source":"myOtherSource", "fieldA":"valueA",
"fieldB":"valueB"}'
```

Splunk HEC Event Endpoint

```
curl -k http://<myCriblHost>:10080/services/collector/event -H 'Authorization:
myToken42' -d '{"event":"this is a sample event ", "host":"myHost",
"source":"mySource", "fieldA":"valueA", "fieldB":"valueB"}'

curl -k http://<myCriblHost>:10080/services/collector -H 'Authorization: myToken42'
-d '{"event":"this is a sample event ", "host":"myHost", "source":"mySource",
"fieldA":"valueA", "fieldB":"valueB"}'
```

> 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 Endpoint** and port, and the token's value:

```
curl -k https://in.main-default-<Your-Org-ID>.cribl.cloud:10080/cribl/_bulk -H
'Authorization: <token_value>' -d '{"_raw":"this is a sample event ",
"host":"myHost", "source":"mySource", "fieldA":"valueA", "fieldB":"valueB"}'
```

9.11. Raw HTTP/S

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.

Configuring Cribl Edge to Receive Raw HTTP Data

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Source** at left. From the resulting drawer's tiles, select [**Pull** >] **Raw HTTP**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). From the resulting page's tiles or left nav, select [**Pull** >] **Raw HTTP**. Next, click **New Source** to open a **New Source** modal that provides the options below.

General Settings

Input ID: Enter a unique name to identify this Raw HTTP Source definition.

Address: Enter the address to bind on. Defaults to `0.0.0.0` (all addresses).

Port: Enter the port number to listen on.

Authentication Settings

Auth tokens: Shared secrets to be provided by any client. Click **Generate** to create a new secret. If empty, permits open access.

Optional Settings

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Edge. Use a tab or hard return between (arbitrary) tag names.

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 (e.g., the system's CA). Defaults to `Yes`.
- **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. E.g., 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

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 Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Enable Persistent Queue** toggle. If enabled, PQ is automatically configured in `Always On` mode,

> with a maximum queue size of 1 GB disk space allocated per Worker Process.

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 Source. Defaults to `1000`.

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, Cribl Edge will stop queueing data, and will apply the **Queue-full behavior**. Enter a numeral with units of KB, MB, etc. If not specified, the implicit `0` default will enable Cribl Edge to fill all available disk space on the volume.

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.

Advanced Settings

Enable Proxy Protocol: Toggle to `Yes` if the connection is proxied by a device that supports Proxy Protocol v1 or v2. This setting affects how the Source handles the `__srcIpPort` field.

Allowed URI paths: List of URI paths accepted by this input. Supports wildcards, e.g., `/api/v*/hook`. Defaults to `*`, which allows all paths.

Allowed HTTP methods: List of HTTP methods accepted by this input. Supports wildcards, e.g., `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.

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.

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.

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** > **Enable proxy protocol** is set to `No`, the original client IP/port in this header will override the value of `__srcIpPort`.

If **Enable proxy protocol** 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.)

9.12. Metrics

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.

Configuring Cribl Edge to Receive Metrics

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Source** at left. From the resulting drawer's tiles, select [**Push** >] **Metrics**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). From the resulting page's tiles or left nav, select [[**Push** >] **Metrics**. Next, click **New Source** to open a **New Source** modal that provides the options below.

General Settings

Input ID: Enter a unique name to identify this Source definition.

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.

TCP port: Enter the TCP port number to listen on. Not required if listening on UDP.

Optional Settings

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Edge. Use a tab or hard return between (arbitrary) tag names.

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 (e.g., the system's CA). Defaults to `Yes`.
- **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. E.g., 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

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 Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Enable Persistent Queue** toggle. If enabled, PQ is automatically configured in `Always On` mode, with a maximum queue size of 1 GB disk space allocated per Worker Process.

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 Source. Defaults to `1000`.

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, Cribl Edge will stop queueing data, and will apply the **Queue-full behavior**. Enter a numeral with units of KB, MB, etc. If not specified, the implicit `0` default will enable Cribl Edge to fill all available disk space on the volume.

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

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`.

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.

9.13. OpenTelemetry (OTel)

Cribl Edge supports receiving trace and metric events from OTLP-compliant senders. (Cribl plans to add support for log events as more components of the OpenTelemetry protocol's logs support graduate to stable status.)

> 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**.

Configuring an OTel Source

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Source** at left. From the resulting drawer's tiles, select [**Push** >] **OpenTelemetry**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). From the resulting page's tiles or left nav, select [**Push** >] **OpenTelemetry**. Next, click **New Source** to open a **New Source** modal that provides the options below.

> The sections described below are spread across several tabs. Click the tab links at left, or the **Next** and **Prev** buttons, to navigate among tabs. Click **Save** when you've configured your Source.

General Settings

Input ID: Unique ID for this Source. E.g., `OTel042`.

Address: Enter the hostname/IP to listen on. Defaults to `0.0.0.0` (all addresses, IPv4 format). IPv6 addresses must be enclosed in square brackets.

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.

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 for filtering and grouping in Cribl Edge. Use a tab or hard return between (arbitrary) tag names.

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. 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 (e.g., the system's CA). Defaults to `Yes`.

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 Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Enable Persistent Queue** toggle. If enabled, PQ is automatically configured in `Always On` mode, with a maximum queue size of 1 GB disk space allocated per Worker Process.

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 Source. Defaults to `1000`.

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, Cribl Edge will stop queueing data, and will apply the **Queue-full behavior**. Enter a numeral with units of KB, MB, etc. If not specified, the implicit `0` default will enable Cribl Edge to fill all available disk space on the volume.

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

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

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.)

The **Extract spans** and **Extract metrics** 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.

Max active requests: Maximum number of active requests allowed for this Source, per Worker Process. Defaults to `256`. Enter `0` for unlimited.

Environment: Optionally, specify a single Git branch on which to enable this configuration. If this field is empty, the config will be enabled everywhere.

When **General Settings** > **Protocol** is set to `HTTP`, the UI displays three additional settings:

- **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).

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
   -days 420
   ```

   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
```

9.14. SNMP Trap

Cribl Edge supports receiving data from SNMP Traps.

> Type: **Push** | TLS Support: **NO** | Event Breaker Support: **No**

Configuring Cribl Edge to Receive SNMP Traps

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Source** at left. From the resulting drawer's tiles, select [**Push** >] **SNMP Trap**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). From the resulting page's tiles or left nav, select [**Push** >] **SNMP Trap**. Next, click **New Source** to open a **New Source** modal that provides the options below.

> Cribl Edge 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.

General Settings

Input ID: Enter a unique name to identify this Source definition.

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`.

Optional Settings

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Edge. Use a tab or hard return between (arbitrary) tag names.

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 Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Enable Persistent Queue** toggle. If enabled, PQ is automatically configured in `Always On` mode, with a maximum queue size of 1 GB disk space allocated per Worker Process.

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 Source. Defaults to `1000`.

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, Cribl Edge will stop queueing data, and will apply the **Queue-full behavior**. Enter a numeral with units of KB, MB, etc. If not specified, the implicit `0` default will enable Cribl Edge to fill all available disk space on the volume.

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

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 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`.

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`
- `__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};`
- `__snmpVersion` : Acceptable values are `0` , `2` , or `3` . These respectively indicate SNMP v1, v2c, and v3.
- `__snmpRaw` : Buffer containing Raw SNMP packet

Considerations for Working with SNMP Trap Data

- It's possible to work with SNMP metadata (i.e., we'll decode the packet). Options include dropping, routing, 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.
- SNMP packets can be forwarded to non-SNMP destinations (e.g., Splunk, Syslog, S3, etc.).
- Non-SNMP input data **cannot** be sent to SNMP destinations.

9.15. Syslog

Cribl Edge supports receiving syslog data, whether structured according to RFC 3164 or RFC 5424. This Source supports message-length prefixes according to RFC 5425 or RFC 6587.

> Type: **Push** | TLS Support: **YES** | Event Breaker Support: **No**
>
> For details on how to replace your syslog server with Cribl Edge, see Syslog Best Practices.

Configuring Cribl Edge to Receive Data over Syslog

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Source** at left. From the resulting drawer's tiles, select [**Push** >] **Syslog**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). From the resulting page's tiles or left nav, select [**Push** >] **Syslog**. Next, click **New Source** to open a **New Source** modal that provides the options below.

> 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.

General Settings

Input ID: Enter a unique name to identify this Syslog Source definition.

Address: Enter the hostname/IP on which to listen for data., E.g. `localhost` or `0.0.0.0`.

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.

TCP port: Enter the TCP port number to listen on. Not required if listening on UDP.

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 for filtering and grouping in Cribl Edge. Use a tab or hard return between (arbitrary) tag names.

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 (e.g., the system's CA). Defaults to `Yes`.
- **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. E.g., 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

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 Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Enable Persistent Queue** toggle. If enabled, PQ is automatically configured in `Always On` mode, with a maximum queue size of 1 GB disk space allocated per Worker Process.

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 Source. Defaults to `1000`.

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, Cribl Edge will stop queueing data, and will apply the **Queue-full behavior**. Enter a numeral with units of KB, MB, etc. If not specified, the implicit `0` default will enable Cribl Edge to fill all available disk space on the volume.

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

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.

Octet count framing: Toggle to `Yes` if messages are prefixed with a byte length, according to RFC 5425 or RFC 6587. The default setting (`No`) applies non-transparent framing using `\n` as the delimiter. The framing method utilized by this input will be applied to all events received by this input. Therefore, if your syslog devices use a mixture of framing types (non-transparent vs. octet count), you will need to use a separate Syslog Source for each framing type. Additional inputs will necessitate separate ports.

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.

IP whitelist 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. 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.

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.

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|tcpp>:<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	
`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|tcpp>:<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.

Troubleshooting Resources

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, click 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.

9.16. TCP JSON

Cribl Edge can receive newline-delimited JSON data over TCP.

> Type: **Push** | TLS Support: **YES** | Event Breaker Support: **No**

Configuring Cribl Edge to Receive TCP JSON Data

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Source** at left. From the resulting drawer's tiles, select [**Push** >] **TCP JSON**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). From the resulting page's tiles or left nav, select [**Push** >] **TCP JSON**. Next, click **New Source** to open a **New Source** modal that provides the options below.

> 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.

General Settings

Input ID: Enter a unique name to identify this TCP JSON Source definition.

Address: Enter hostname/IP to listen for TCP JSON data. E.g., `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 Edge'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 Cribl Edge. Use a tab or hard return between (arbitrary) tag names.

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 (e.g., the system's CA). Defaults to `Yes`.

- **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. E.g., 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

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 Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Enable Persistent Queue** toggle. If enabled, PQ is automatically configured in `Always On` mode, with a maximum queue size of 1 GB disk space allocated per Worker Process.

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 Source. Defaults to `1000`.

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, Cribl Edge will stop queueing data, and will apply the **Queue-full behavior**. Enter a numeral with units of KB, MB, etc. If not specified, the implicit `0` default will enable Cribl Edge to fill all available disk space on the volume.

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

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. 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.

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 – e.g., `{}`. 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":"valueA", "fieldB":"valueB"}
{"host":"myOtherHost", "source":"myOtherSource", "_raw": "
{\"message\":\"Something informative happened\", \"severity\":\"INFO\"}"}
```

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: `in.main-default-<Your-Org-ID>.cribl.cloud` – you can simply copy/paste your Cribl Cloud portal's **Ingest Endpoint** here

Port: `10070`

TLS Settings (Client Side)

Enabled: `Yes`

Validate server certs: `Yes`

9.17. TCP (Raw)

Cribl Edge supports receiving of data over TCP. (See examples and header options below.)

> Type: **Push** | TLS Support: **YES** | Event Breaker Support: **YES**

Configuring Cribl Edge to Receive TCP Data

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Source** at left. From the resulting drawer's tiles, select [**Push** >] **TCP**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). From the resulting page's tiles or left nav, select [**Push** >] **TCP**. Next, click **New Source** to open a **New Source** modal that provides the options below.

> 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.

General Settings

Input ID: Enter a unique name to identify this TCP Source definition.

Address: Enter hostname/IP to listen for raw TCP data. E.g., `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, etc. 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.

Optional Settings

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Edge. Use a tab or hard return between (arbitrary) tag names.

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 (e.g., the system's CA). Defaults to `Yes`.
- **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. E.g., 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

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 Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Enable Persistent Queue** toggle. If enabled, PQ is automatically configured in `Always On` mode, with a maximum queue size of 1 GB disk space allocated per Worker Process.

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 Source. Defaults to `1000`.

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, Cribl Edge will stop queueing data, and will apply the **Queue-full behavior**. Enter a numeral with units of KB, MB, etc. If not specified, the implicit `0` default will enable Cribl Edge to fill all available disk space on the volume.

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

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.

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`:

```
cat test.raw | nc --ssl --ssl-verify in.main-default-<Your-Org-ID>.cribl.cloud 10060
```

9.18. UDP (Raw)

Cribl Edge supports receiving raw, unparsed data via UDP.

> Type: **Push** | TLS Support: **NO** | Event Breaker Support: **NO**

Configuring Cribl Edge to Receive Raw UDP Data

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **+ Add Source** at left. From the resulting drawer's tiles, select [**Push** >] **Raw UDP**. Next, click either **+ Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). From the resulting page's tiles or left nav, select [**Push** >] **Raw UDP**. Next, click **Add Source** to open a **New Source** modal that provides the options below.

General Settings

Input ID: Enter a unique name to identify this raw UDP Source definition.

Address: Enter the hostname/IP to listen for raw UDP data. E.g., `localhost`, `0.0.0.0`, or `::` (for IPv6).

Port: Enter the port number.

Optional Settings

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Edge. Use a tab or hard return between (arbitrary) tag names.

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 Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Enable Persistent Queue** toggle. If enabled, PQ is automatically configured in `Always On` mode, with a maximum queue size of 1 GB disk space allocated per Worker Process.

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 Source. Defaults to `1000`.

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, Cribl Edge will stop queueing data, and will apply the **Queue-full behavior**. Enter a numeral with units of KB, MB, etc. If not specified, the implicit `0` default will enable Cribl Edge to fill all available disk space on the volume.

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

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.

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

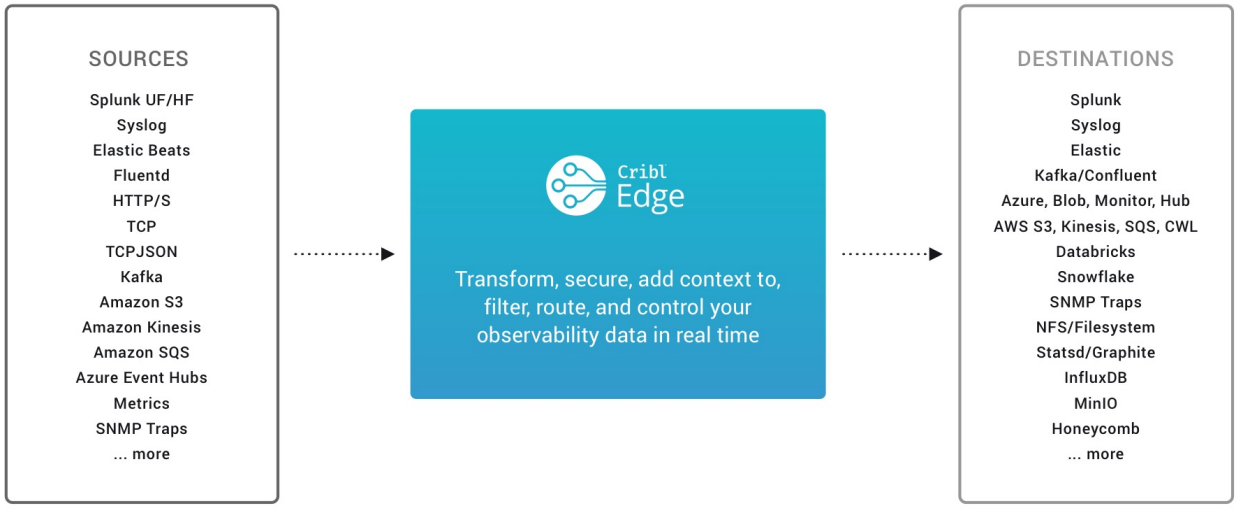
Cribl Edge uses a set of internal fields to assist in handling of data. These "meta" fields are **not** part of an event, but they are accessible, and functions can use them to make processing decisions.

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.

10. Destinations

Cribl Edge can send data to various Destinations, including Splunk, Kafka, Kinesis, InfluxDB, Snowflake, Databricks, TCP JSON, and many others. Destinations can write data to either IPv4 or IPv6 addresses.



Destinations in the Edge ecosystem

Streaming Destinations

Destinations that accept events in real time are referred to as streaming Destinations:

- Amazon CloudWatch Logs
- Amazon Kinesis Streams
- Amazon MSK
- Amazon SQS
- Azure Event Hubs
- Azure Monitor Logs
- Azure Sentinel
- Confluent Cloud
- Cribl HTTP
- Cribl TCP
- CrowdStrike Falcon LogScale
- Datadog
- DataSet

- Elasticsearch
- Google Chronicle
- Google Cloud Logging
- Google Cloud Pub/Sub
- Grafana Cloud
- Graphite
- Honeycomb
- InfluxDB
- Kafka
- Loki
- New Relic Events
- New Relic Logs & Metrics
- OpenTelemetry (OTel)
- Prometheus
- SignalFx
- SNMP Trap
- Splunk HEC
- Splunk Load Balanced
- Splunk Single Instance
- StatsD
- StatsD Extended
- Sumo Logic
- Syslog
- TCP JSON
- Wavefront
- Webhook

Non-Streaming Destinations

Destinations that accept events in groups or batches are referred to as non-streaming Destinations:

- Amazon S3 Compatible Stores
- Data Lakes > Amazon S3
- Data Lakes > Amazon Security Lake
- Azure Blob Storage

- Filesystem/NFS
- Google Cloud Storage
- MinIO

> The Amazon S3 Compatible Stores Destination can be adapted 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 and Special-Purpose Destinations

These special-purpose Destinations route data within your Cribl Edge deployment, or among Workers across distributed or hybrid Cloud deployments:

- 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.
- Cribl TCP: Send data among peer Edge Nodes over TCP.
- Cribl Stream (Deprecated): Use either Cribl HTTP or Cribl TCP instead.
- **SpaceOut**: This experimental Destination is undocumented. Be careful!

How Does Non-Streaming Delivery Work

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 – typically file size and number of files, further details below – data is compressed and then moved to the final Destination.

An inventory of open, or in-progress, files is kept in the staging directory's root, to avoid having to walk that directory at startup. This can get expensive if staging is also the final directory. 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

Several **conditions** govern when files are closed and rolled out:

1. File reaches its configured maximum size.
2. File reaches its configured maximum open time.
3. File reaches its configured maximum 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.

Data Delivery and Persistent Queues

Cribl Stream attempts to deliver data to all Destinations on an at-least-once basis. When a Destination is unreachable, there are three possible behaviors:

- **Block** - Cribl Stream will block incoming events.
- **Drop** - Cribl Stream will drop events addressed to that Destination.
- **Queue** - To prevent data loss, Cribl Stream will write events to a **Persistent Queue** disk buffer, then forward them when a Destination becomes available. (Available on several streaming Destinations.)

You can configure your desired behavior through a Destination's **Backpressure Behavior** drop-down. Where other options are not displayed, Cribl Stream's default behavior is **Block**. For details about all the above behaviors and options, see Persistent Queues.

Configuring Destinations

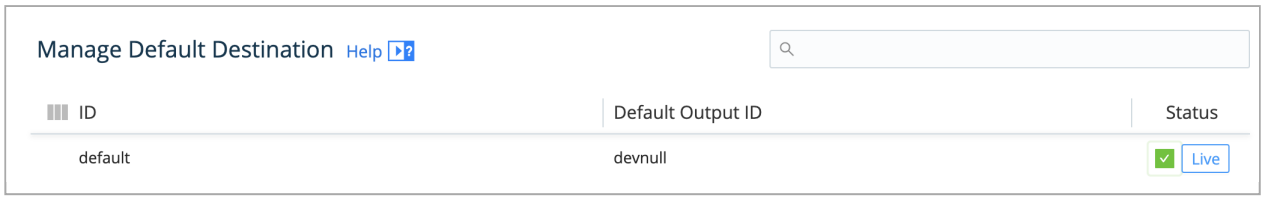
For each Destination **type**, you can create multiple definitions, depending on your requirements.

To configure Destinations, from the top nav of a distributed deployment, first click **Manage**, then select a **Fleet** to configure and choose one of the options below. For a single-instance deployment proceed to the options below:

- To access the graphical QuickConnect UI, click **Collect**. Next, select the desired type, and then click either **Add New** or (if displayed) **Select Existing**.
- To access the Routing UI, click **More** > **Destinations**. On the resulting **Manage Destinations** page's tiles, select the desired type, then click **New Destination**.

Capturing Outgoing Data

To capture data from a single enabled Destination, you can bypass the Preview pane, and instead capture directly from a **Manage Destinations** page. Just click 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 clicking the modal's **Live Data** tab.

Destination modal > Live Data tab

> Cribl University offers a course titled Sending Data with Edge that provides an illustrated overview. To follow the direct course link, first log into your Cribl University account. (To create an account, click 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.

10.1. Internal

10.1.1. Cribl HTTP

The Cribl HTTP Destination is available only in distributed deployments. It enables Edge Nodes to send data to peer Nodes, as long as all Nodes are connected to the same Leader.

You'll find the Cribl HTTP Destination/Source pair especially valuable in a hybrid Cloud deployment, where it ensures that you're billed for ingress only once – when Cribl first receives the data. All data subsequently transferred to other Workers via the Cribl HTTP Destination/Source pair is not charged. 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.

In single-instance mode, or for testing, you can substitute the Webhook Destination. (However, this substitution will not facilitate sending all internal fields, as described below.)

> Type: Streaming | TLS Support: Configurable | PQ Support: Yes

You might choose this Destination over the Cribl TCP Destination in certain circumstances, such as when a firewall or proxy blocks raw TCP egress.

Configuration Requirements

Configuring Cribl HTTP flow between peer Edge Nodes imposes some particular requirements:

- The Cribl HTTP Destination must be on a Edge Node that is connected to the same Leader as the Cribl HTTP Source(s).
- You must specify the same Leader Address on the Edge Nodes that host both the Destination and Source. Otherwise, token verification will fail – breaking the connection and preventing data flow.
- To get the Leader Address specifically for Cribl.Cloud hybrid Workers, see Hybrid Cribl HTTP/Cribl TCP Configuration.
- To configure the Leader Address via the UI, log directly into each Edge Node's UI. Then select **Settings** > **Global Settings** > **Distributed Settings** > **Leader Settings** > **Address**.

- To configure the Leader Address via the instance.yml file, the `host` values on the connecting Edge Nodes must be identical. In this example, both Edge Nodes must point to `cribl-leader`:

```
distributed:
  mode: master
  master:
    host: cribl-leader
    port: 4200
```

- This Destination's **Cribl endpoint** field must point to the **Address** and **Port** you've configured on its peer Cribl HTTP Source(s).

Configuring a Cribl HTTP Destination

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **Cribl HTTP**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **Cribl HTTP**. Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this Destination definition.

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, e.g., `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.

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 for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Load Balancing Settings

Enabling the **Load balancing** slider displays the following controls.

Exclude Current Host IPs: This slider 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 each connection's relative traffic-handling capability 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 section displays 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 the queue is allowed to consume. Once this limit is reached, Cribl Stream 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 Stream 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 button if you want to flush out files that are currently queued for delivery to this Destination. A confirmation modal will appear. (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 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

Validate server certs: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (e.g., 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 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. Defaults to `4096` KB.

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. By default, `__kube_*` and `__metadata` are excluded. This Destination forwards all other Internal Fields.

> 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.

10.1.2. Cribl TCP

The Cribl TCP Destination is available only in distributed deployments. It enables Edge Nodes to send data to peer Nodes, as long as all Nodes are connected to the same Leader.

You'll find the Cribl TCP Destination/Source pair especially valuable in a hybrid Cloud deployment, where it ensures that you're billed for ingress only once – when Cribl first receives the data. All data subsequently transferred to other Workers via the Cribl TCP Destination/Source pair is not charged. However, the Cribl TCP Destination/Source pair can similarly reduce your metered data ingress in other scenarios, such as on-prem Edge to on-prem Stream.

> Type: Streaming | TLS Support: Configurable | PQ Support: Yes

You might choose this Destination over the Cribl HTTP Destination in certain circumstances, such as when a firewall or proxy allows raw TCP egress. In single-instance mode or for testing, you can substitute the TCP JSON Destination. (However, this substitution will not facilitate sending all internal fields, as described below.)

Configuration Requirements

Configuring Cribl TCP flow between peer Edge Nodes imposes some particular requirements:

- The Cribl TCP Destination must be on a Edge Node that is connected to the same Leader as the Cribl TCP Source(s).
- You must specify the same Leader Address on the Edge Nodes that host both the Destination and Source. Otherwise, token verification will fail – breaking the connection and preventing data flow.
- To get the Leader Address specifically for Cribl.Cloud hybrid Workers, see Hybrid Cribl HTTP/Cribl TCP Configuration.
- To configure the Leader Address via the UI, log directly into each Edge Node's UI. Then select **Settings** > **Global Settings** > **Distributed Settings** > **Leader Settings** > **Address**.
- To configure the Leader Address via the instance.yml file, the `host` values on the connecting Edge Nodes must be identical. In this example, both Edge Nodes must point to `cribl-leader`:

```
distributed:
  mode: master
  master:
    host: cribl-leader
    port: 4200
```

- This Destination's **Address** and **Port** values must match the **Address** and **Port** you've configured on its peer Cribl TCP Source(s).

Configuring a Cribl TCP Destination

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **Cribl TCP**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **Cribl TCP**. Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this Destination definition.

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, e.g., `10300`.

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, e.g., `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 for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Load Balancing Settings

Enabling the **Load balancing** slider displays the following controls:

Exclude Current Host IPs

This slider 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 Cribl HTTP 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 each connection's relative traffic-handling capability 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 section displays 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 the queue is allowed to consume. Once this limit is reached, Cribl Stream 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 Stream 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 button if you want to flush out files that are currently queued for delivery to this Destination. A confirmation modal will appear. (Appears only after **Output ID** has been defined.)

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 (e.g., 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. By default, `__kube_*` and `__metadata` are excluded. This Destination forwards all other Internal Fields.

> 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.

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.

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`.

10.1.3. Cribl Stream (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.

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, e.g., `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 for filtering and grouping at the final destination. 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 Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Clear Persistent Queue** button. A maximum queue size of 1 GB disk space is automatically allocated

> per Worker Process. If the queue fills up, Cribl Edge will block outbound data.

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.

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 button if you want to flush out files that are currently queued for delivery to this Destination. A confirmation modal will appear. (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 (e.g., 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.

10.1.4. Default

The **Default** Destination simply enables you to specify a default output from among your already-configured Destinations.

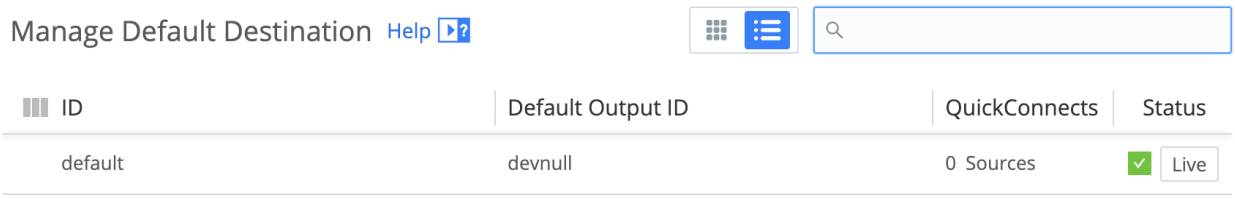
> Type: Internal | TLS Support: N/A | PQ Support: N/A

Configuring Cribl Edge's Default Destination

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). 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.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **Default**. From the resulting **Manage Default Destination** page, click anywhere on the `default` row to proceed.



Default Destination – click to configure

In the resulting **Destinations > Default** drawer or modal, use the **Default Output ID** drop-down to select one of your configured Destinations. After you click **Save**, this will become Cribl Edge's default Destination.

The only other field here is the **Output ID**, whose value is locked to `default`.

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.

10.1.5. DevNull

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

Configuring 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, from the top nav, click **Manage**, then select a **Fleet** to configure. Next, select **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **Devnull**. Look for the **Live** indicator at the top right.

In the QuickConnect UI, click **Collect** (Edge only), a **DevNull** Destination is preconfigured and ready to use in the right **Destinations** column.

10.1.6. Output Router

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

Configuring Cribl Edge to Send to an Output Router

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **Output Router**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **Output Router**.
Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Router name: Enter a unique name to identify this Router definition.

Rules: A list of event routing rules. Each provides the following 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.

Optional 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.

Tags: Optionally, add tags that you can use for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

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

FILTER EXPRESSION	OUTPUT	FINAL
`method=='POST' || method=='GET`	**Seattle**	No
`method=='POST' || method=='GET'`	**Los Angeles**	Yes
`true`	**New York**	Yes

10.2. Amazon

10.2.1. Amazon CloudWatch Logs

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

Configuring Cribl Edge to Output to Amazon CloudWatch Logs

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **Amazon** > **CloudWatch Logs**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **Amazon** > **CloudWatch Logs**. Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this CloudWatch definition.

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. (E.g., `myStream_myHost_myOutputId`.)

Region: AWS region where the CloudWatch Logs group is located.

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 for filtering and grouping at the final destination. 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 Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Clear Persistent Queue** button. A maximum queue size of 1 GB disk space is automatically allocated per Worker Process. If the queue fills up, Cribl Edge will block outbound data.

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.

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 button if you want to flush out files that are currently queued for delivery to this Destination. A confirmation modal will appear. (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.

Authentication

Use the **Authentication Method** buttons to select an AWS authentication method.

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 Edge Workers 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 Workers not in an AWS VPC, e.g., 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

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.

Signature version: Signature version to use for signing CloudWatch Logs requests. Defaults to `v4`.

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 (e.g., 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.

10.2.2. Amazon Kinesis Streams

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

Configuring Cribl Edge to Output to Amazon Kinesis Data Streams

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **Amazon** > **Kinesis**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **Amazon** > **Kinesis**.
Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this Kinesis definition.

Stream name: Enter the name of the Kinesis Data Stream to which to send events.

Region: Select the AWS Region where the Kinesis Data Stream is located.

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 for filtering and grouping at the final destination. 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 Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Clear Persistent Queue** button. A maximum queue size of 1 GB disk space is automatically allocated per Worker Process. If the queue fills up, Cribl Edge will block outbound data.

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.

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** drops the newest events from being sent out of Cribl Edge, and throws away incoming data, while leaving the contents of the PQ unchanged.

Clear persistent queue: Click this button if you want to flush out files that are currently queued for delivery to this Destination. A confirmation modal will appear. (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.

Authentication

Use the **Authentication Method** buttons to select an AWS authentication method.

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 Edge Workers 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 Workers not in an AWS VPC, e.g., 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

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 (e.g., 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.

Format

Currently, outputted events use the following record format:

- Header line containing information about the payload (currently supports one type, as shown below).
- Newline-Delimited JSON (that is, each Kinesis record will contain multiple events, in **ndjson** format).

Record payloads (including header and body) will be gzip-compressed, and then Kinesis will base64-encode them.

Sample Kinesis Record


```
{"format":"ndjson","count":8,"size":3960}
{"_raw":"07-03-2018 18:33:51.136 -0700 ERROR TcpOutputFd - Read error. Connection
reset by peer","_meta":"timestartpos::0 timeendpos::29 _subsecond::.136
date_second::51 date_hour::18 date_minute::33 date_year::2018 date_month::july
date_mday::3 date_wday::tuesday date_zone::-420 punct::--_::._-___-
__.____","_time":"1530668031","source":"/mnt-
big/ledion/hwf/var/log/foo/food.log","host":"ledion-
hwf","sourcetype":"food","index":"_internal","cribl_pipe":"foo2"}
{"_raw":"07-03-2018 18:33:51.136 -0700 INFO  TcpOutputProc - Connection to
127.0.0.1:10000 closed. Read error. Connection reset by
peer","_meta":"timestartpos::0 timeendpos::29 _subsecond::.136 date_second::51
date_hour::18 date_minute::33 date_year::2018 date_month::july date_mday::3
date_wday::tuesday date_zone::-420 punct::--_::._-____-
___...:_.__.____","_time":"1530668031","source":"/mnt-
big/ledion/hwf/var/log/foo/food.log","host":"ledion-
hwf","sourcetype":"food","index":"_internal","cribl_pipe":"foo2"}
...
```

10.2.3. Amazon S3 Compatible Stores

S3 is a non-streaming Destination type. Cribl Edge does **not** have to run on AWS in order to deliver data to S3.

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 these integrations, please contact Cribl Support.

> Type: Non-Streaming | TLS Support: Yes | PQ Support: No

Configuring Cribl Edge to Output to S3 Destinations

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **Amazon** > **S3**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **Amazon** > **S3**. Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this S3 definition.

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. E.g., 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. (This 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.

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. File name prefix and suffix expressions do not bypass this behavior.
>
> 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 for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Authentication

Use the **Authentication Method** buttons to select one of these options:

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 Edge Workers 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 Workers not in an AWS VPC, e.g., 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. 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

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:

- 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.
- The S3 Collector currently does not support ingesting data in the Parquet format. Therefore, data that you export in Parquet format cannot be replayed via the Collector.

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.

Row group size: Set the target memory size for row group segments. Modify this value to optimize memory use when writing. Value must be a positive integer smaller than the **File size** value, with appropriate units. Defaults to `16 MB`.

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`.

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`.

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.

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.

Max concurrent file parts: Maximum number of parts to upload in parallel per file. Minimun part size is 5 MB. A value of `1` will upload the file's contents sequentially, with a maximum concurrency of `1`. Defaults to `4`.

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 slider'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 to `Yes` 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).

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`; 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`.

> 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 (e.g., 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`.

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.

Amazon S3 Permissions

The following permissions are needed to write to an Amazon S3 bucket:

- `s3:ListBucket`
- `s3:GetBucketLocation`
- `s3:PutObject`

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 Resources

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, click 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.

See also AWS Sources/Destinations & S3-Compatible Stores for information on common errors.

10.2.4. Amazon SQS

Cribl Edge supports sending events to Amazon Simple Queuing Service.

> Type: Streaming | TLS Support: Yes | PQ Support: Yes

Configuring Cribl Edge to Send Data to Amazon SQS

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **Amazon** > **SQS**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **Amazon** > **SQS**.
Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this SQS Destination.

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>'`. (E.g., `':port/<myQueueName>'`.)

Queue type: The queue type used (or created). Defaults to `Standard`. `FIFO` (First In, First Out) is the other option.

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`.

Tags: Optionally, add tags that you can use for filtering and grouping at the final destination. 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 Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Clear Persistent Queue** button. A maximum queue size of 1 GB disk space is automatically allocated per Worker Process. If the queue fills up, Cribl Edge will block outbound data.

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.

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 button if you want to flush out files that are currently queued for delivery to this Destination. A confirmation modal will appear. (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.

Authentication

Use the **Authentication Method** buttons to select an AWS authentication method.

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 Edge Workers 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 Workers not in an AWS VPC, e.g., 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

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 (e.g., 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.

10.2.5. Amazon Security Lake

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.

> 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 shown 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. Meanwhile, refer to the present topic as needed for details about any given setting.

Although Cribl Edge does **not** need to be deployed in Cribl.Cloud in order to deliver data to Amazon Security Lake, setup procedures will differ from those documented here. To learn more, please contact Cribl via the `#packs` channel of Cribl Community Slack.

Configuring Cribl Edge to Output to Amazon Security Lake

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **Data Lakes** > **Amazon Security Lake**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **Data Lakes** > **Amazon Security Lake**. Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

> The Integration topic should be your primary guide to setting up Amazon Security Lake. For that reason, some of the instructions below assume that you have noted IDs, values, etc., from procedures you have already completed as directed by the Integration topic.

Output ID: Enter a unique name to identify this Amazon Security Lake Destination definition.

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. E.g., 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. (This field is not displayed or available on Cribl.Cloud-managed Edge Nodes.)

Optional Settings

File name prefix expression: The output filename prefix. Must be a JavaScript expression (which can evaluate to a constant), enclosed in quotes or backticks. Defaults to `CriblOut`.

Backpressure behavior: Select whether to block or drop events when all receivers are exerting backpressure. (Causes might include an accumulation of too many files needing to be closed.) Defaults to `Block`.

Tags: Optionally, add tags that you can use for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

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

Note that:

- Cribl Edge Workers support Parquet only when running on Linux, not on Windows.

Row group size: Ideal memory size for row group segments. E.g., 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. E.g., 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.

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.

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 to `Yes` 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).

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 (e.g., 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.

Amazon S3 Permissions

The following permissions are needed to write to an Amazon S3 bucket:

- `s3:ListBucket`
- `s3:GetBucketLocation`
- `s3:PutObject`

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, click 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.

10.3. Azure

10.3.1. Azure Blob Storage

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

Configuring Cribl Edge to Output to Azure Blob Storage

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **Azure** > **Blob Storage**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **Azure** > **Blob Storage**. Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this Destination definition.

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. (This 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.

Authentication Settings

Use the **Authentication method** buttons to select one of these options:

- **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.

Connection String Format

Either authentication method uses an Azure Storage connection string in this format:

```
DefaultEndpointsProtocol=[http|https];AccountName=<your-account-name>;AccountKey=
<your-account-key>;EndpointSuffix=<your-endpoint-suffix>
```

A fictitious example, using Microsoft's recommended HTTPS option, is:

```
DefaultEndpointsProtocol=https;AccountName=storagesample;AccountKey=12345678...32;Endpo
```

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. File name prefix and suffix expressions do not bypass this behavior.
>
> 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 for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

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:

- 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.

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.

Row group size: Set the target memory size for row group segments. Modify this value to optimize memory use when writing. Value must be a positive integer smaller than the **File size** value, with appropriate units. Defaults to `16 MB`.

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`.

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`.

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.

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.

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 slider'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 to `Yes` 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).

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.

10.3.2. Azure Event Hubs

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.

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

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **Azure** > **Events Hub**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **Azure** > **Events Hub**.
Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this Azure Event Hubs definition.

Brokers: List of Event Hub Kafka brokers to connect to. (E.g., `yourdomain.servicebus.windows.net:9093`.) Find the hostname in Shared Access Policies, in the host portion of the primary or secondary connection string.

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.

Optional Settings

Acknowledgments: Control the number of required acknowledgments. Defaults to `Leader`.

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 for filtering and grouping at the final destination. 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 Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Clear Persistent Queue** button. A maximum queue size of 1 GB disk space is automatically allocated per Worker Process. If the queue fills up, Cribl Edge will block outbound data.

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.

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 button if you want to flush out files that are currently queued for delivery to this Destination. A confirmation modal will appear. (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 `Yes`.

Validate server certs: Defaults to `No` – and for Event Hubs, this must always be disabled.

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 `OAUTHBREAKER` 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=DummyAccessKeyName;SharedAccessKey=5dOntTRytoC24opYThisAsit3is2B+OGY1US/fuL3ly=
```

OAuth Authentication

Selecting the `OAUTHBREAKER` SASL mechanism provides the options listed in this section.

Azure AD authentication endpoint: Specifies the Azure Active Directory 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 Azure Active Directory 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`. (E.g., 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.

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 (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, i.e., 10 seconds. Valid range is `1000` to `3600000` ms, i.e., 1 second to 1 hour.

Request timeout (ms): Maximum time to wait for a successful request. Defaults to `60000` ms, i.e., 1 minute.

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.

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:

- `__topicOut`

- `__key`
- `__headers`
- `__keySchemaIdOut`
- `__valueSchemaIdOut`

Troubleshooting Resources

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, click 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.

10.3.3. Azure Monitor Logs

Cribl Edge supports sending data to Azure Monitor Logs.

> Type: Streaming | TLS Support: Yes | PQ Support: Yes

Configuring Cribl Edge to Output to Azure Monitor Logs

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **Azure** > **Monitor Logs**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **Azure** > **Monitor Logs**. Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this Azure Monitor Logs definition.

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.

Authentication Settings

Authentication method: Use the buttons to select one of these options:

- **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.

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 for filtering and grouping at the final destination. 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 Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Clear Persistent Queue** button. A maximum queue size of 1 GB disk space is automatically allocated per Worker Process. If the queue fills up, Cribl Edge will block outbound data.

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.

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 button if you want to flush out files that are currently queued for delivery to this Destination. A confirmation modal will appear. (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.

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.

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. Defaults to `4096`.

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 2 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.

10.3.4. Azure Sentinel

Cribl Edge can send log and metric events to the Azure Sentinel SIEM. (These docs use the terms "Azure Sentinel" and "Microsoft Sentinel" interchangeably.)

> Type: Streaming | TLS Support: No | PQ Support: Yes

Configuring Cribl Edge to Output to Azure Sentinel

> See Azure Sentinel SIEM Integration for the overall procedure into which this topic's instructions fit.

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **Azure Sentinel**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **Azure Sentinel**.
Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this HTTP output definition.

Endpoint configuration: Method for configuring the endpoint. Options include:

- **URL** – lets you directly enter the data collection endpoint.
- **ID** – lets you enter individual IDs.

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 route ID: Immutable ID for the data collection route (DCR).

Stream name: Name of the Sentinel table in which to store events.

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 for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Authentication

- **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.
- **Client ID**: JavaScript expression used to compute the application (client) ID for the Azure application. Can be a constant.

Persistent Queue Settings

> This tab is displayed when the **Backpressure behavior** is set to **Persistent Queue**.
>
> On Cribl-managed Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Clear Persistent Queue** button. A maximum queue size of 1 GB disk space is automatically allocated per Worker Process. If the queue fills up, Cribl Edge will block outbound data.

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.

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.

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

Validate server certs: Reject certificates that are not authorized by a trusted CA (e.g., 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. 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 before blocking. This is set per Worker Process. Defaults to `5`.

Max body size (KB): Maximum size of the request body. Defaults to `1000` 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.

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 Azure 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 Azure 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" }`.

Use Case

See Azure Sentinel Integration for more information on configuring an Azure Sentinel Destination to integrate with the Azure Sentinel SIEM.

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 2 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.

10.4. Google Cloud

10.4.1. Google Chronicle

Cribl Edge supports sending data to Google Chronicle, a cloud service for retaining, analyzing, and searching enterprise security and network telemetry data.

To define a Google Chronicle Destination, you need to obtain an API key from Google. If you want Cribl Edge or an external KMS to manage the API key, configure a key pair that references the API key.

> Type: Streaming | TLS Support: Yes | PQ Support: Yes

Configuring Cribl Edge to Output to Chronicle

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **Google Cloud** > **Chronicle** Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **Google Cloud** > **Chronicle**. Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this Chronicle output definition.

Default log type: Select an application log type to send to Chronicle. (Google Chronicle expects all batches for a given Destination to have the same log type.) Can be overwritten by the `__logType` event field, and interacts with the optional **Custom log types** controls below. See the Google documentation for a current list of supported log types.

Optional Settings

Custom log types: In Cribl Edge 4.0 and later, you can use the **Add type** button here to define each desired type. Each will be a table row with **Log type** and **Description** fields. If you set the **Default log type** drop-down above to the value `Custom`, Cribl Edge will automatically select the first custom log type in this table as the default log type. Use the grab handles at left to reorder table rows.

Namespace: 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 Chronicle instance. To use this optional field, request the ID from your Chronicle representative.

Send events as: `Unstructured` is the only currently supported format. Cribl plans to add UDM (Unified Data Model) support in a future release.

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.

Region: From the drop-down, choose the Google Chronicle 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 for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Authentication

The Google Chronicle API key is required to complete this part of the Destination definition.

Use the **Authentication Method** buttons to select one of these options:

- **Manual**: In the resulting **API key** field, enter your Google Chronicle API key.
- **Secret**: This option exposes a **Secret** drop-down, in which you can select a stored secret that references your Google Chronicle API key. A **Create** link is available to store a new, reusable secret.

Persistent Queue Settings

> This section 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 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`, etc.

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: Click this button if you want to flush out files that are currently queued for delivery to this Destination. A confirmation modal will appear. (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: 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.

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.

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.

Request concurrency: Enter the maximum number of ongoing requests to allow before blocking.

Max body size (KB): Enter a maximum size, in KB, for the request body.

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.

10.4.2. Google Cloud Logging

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

Configuring Cribl Edge to Output to Google Cloud Logging

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **Google Cloud** > **Logging**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **Google Cloud** > **Logging**. Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this Google Cloud Logging definition.

Log location type: Select one of the following.

- `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.

Log name expression: Ener a JavaScript expression to compute the log name's value.

Field Mappings

Payload format: Select one of the following.

- `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.

> 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. Each 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.

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.

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 for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Authentication

Use the **Authentication Method** buttons 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 is displayed when the **Backpressure behavior** is set to **Persistent Queue**.
>
> On Cribl-managed Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Clear Persistent Queue** button. A maximum queue size of 1 GB disk space is automatically allocated per Worker Process. If the queue fills up, Cribl Edge will block outbound data.

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.

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 button if you want to flush out files that are currently queued for delivery to this Destination. A confirmation modal will appear. (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.

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. Defaults to `4096` KB.

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`

10.4.3. Google Cloud Pub/Sub

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

Configuring Cribl Edge to Output to Pub/Sub

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **Google Cloud** > **Pub/Sub**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **Google Cloud** > **Pub/Sub**. Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this Pub/Sub output definition.

Topic ID: ID of the Pub/Sub topic to send events to.

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 this to work correctly, the process receiving events must have ordering 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 for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Authentication

Use the **Authentication Method** buttons to select one of these options:

- **Auto**: This option uses the environment variables `PUBSUB_PROJECT` and `PUBSUB_CREDENTIALS`, 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 is displayed when the **Backpressure behavior** is set to **Persistent Queue**.
>
> On Cribl-managed Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Clear Persistent Queue** button. A maximum queue size of 1 GB disk space is automatically allocated per Worker Process. If the queue fills up, Cribl Edge will block outbound data.

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.

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 button if you want to flush out files that are currently queued for delivery to this Destination. A confirmation modal will appear. (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.

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 (e.g.) 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.

10.4.4. Google Cloud Storage

Google Cloud Storage is a non-streaming Destination type.

> Type: Non-Streaming | TLS Support: Yes | PQ Support: No

Configuring Cloud Storage Permissions

For Cribl Edge to send data to Google Cloud Storage 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.

Configuring Cribl Edge to Output to Cloud Storage Destinations

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles or the **Destinations** left nav, select **Google Cloud** > **Storage**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles, select **Google Cloud** > **Storage**. Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this Cloud Storage definition.

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. E.g., 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. (This field is not displayed or available on Cribl.Cloud-managed Edge Nodes.)

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.

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. File name prefix and suffix expressions do not bypass this behavior.
>
> 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 for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Authentication

Use the **Authentication Method** buttons to select one of these options:

- **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 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:

- 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.

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.

Row group size: Set the target memory size for row group segments. Modify this value to optimize memory use when writing. Value must be a positive integer smaller than the **File size** value, with appropriate units. Defaults to `16 MB`.

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`.

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`.

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.

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`.

Remove staging dirs: Toggle to `Yes` 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).

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 (e.g., 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

Nonspecific messages from Google Cloud of the form `Error: failed to close file` can indicate problems with the permissions listed above.

10.5. Kafka

10.5.1. Kafka

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.

Configuring Cribl Edge to Output to Kafka

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **Kafka**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **Kafka**. Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this Kafka definition.

Brokers: List of Kafka brokers to connect to. (E.g., `localhost:9092`.)

Topic: The topic on which to publish events. Can be overwritten using event's `__topicOut` field.

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`.

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 for filtering and grouping at the final destination. 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 Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Clear Persistent Queue** button. A maximum queue size of 1 GB disk space is automatically allocated per Worker Process. If the queue fills up, Cribl Edge will block outbound data.

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, queueing is stopped and data blocking is applied. Enter a numeral with units of KB, MB, etc.

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. Select `None`, `Gzip`, `Snappy`, or `LZ4`. Defaults to `Gzip`.

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 button if you want to flush out files that are currently queued for delivery to this Destination. A confirmation modal will appear. (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.

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 **Advanced Settings** > **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: 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). 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.

Authentication

This section governs SASL (Simple Authentication and Security Layer) authentication 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 authentication 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

Selecting Kerberos as the authentication mechanism displays the following options:

Keytab location: Enter the location of the key table file for the authentication principal.

Principal: Enter the authentication principal, e.g.: `kafka_user@example.com`.

Broker service class: Enter the Kerberos service class for Kafka brokers, e.g.: `kafka`.

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. (E.g., `http://<hostname>: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.

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 (e.g., 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.

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, i.e., 10 seconds. Valid range is `1000` to `3600000` ms, i.e., 1 second to 1 hour.

Request timeout (ms): Maximum time to wait for a successful request. Defaults to `60000` ms, i.e., 1 minute.

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.

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:

- `__topicOut`
- `__key`
- `__headers`
- `__keySchemaIdOut`
- `__valueSchemaIdOut`

10.5.2. Confluent Cloud

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.

Sending Kafka Topic Data to Confluent Cloud

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **Confluent Cloud**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **Confluent Cloud**.
Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this Destination definition.

Brokers: List of Confluent Cloud brokers to connect to. (E.g., `myAccount.confluent.cloud:9092`.)

Topic: The topic on which to publish events. Can be overwritten using event's `__topicOut` field.

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`.

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 for filtering and grouping at the final destination. 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 Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Clear Persistent Queue** button. A maximum queue size of 1 GB disk space is automatically allocated per Worker Process. If the queue fills up, Cribl Edge will block outbound data.

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, queueing is stopped and data blocking is applied. Enter a numeral with units of KB, MB, etc.

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`, `Snappy`, and `LZ4` are also available.

> Cribl strongly recommends enabling compression. Doing so improves Cribl Edge's performance, enabling faster data transfer using less bandwidth.

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 button if you want to flush out files that are currently queued for delivery to this Destination. A confirmation modal will appear. (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.

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 **Advanced Settings** > **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 (e.g., the system's CA).

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.

Authentication

Authentication parameters to use when connecting to brokers. 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` and `SCRAM-SHA-512` are also available.

Use the **Authentication method** buttons to select one of these options:

- **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.

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. (E.g., `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.

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 (e.g., 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.

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, i.e., 10 seconds. Valid range is `1000` to `3600000` ms, i.e., 1 second to 1 hour.

Request timeout (ms): Maximum time to wait for a successful request. Defaults to `60000` ms, i.e., 1 minute.

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.

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:

- `__topicOut`
- `__key`
- `__headers`
- `__keySchemaIdOut`
- `__valueSchemaIdOut`

10.5.3. Amazon MSK

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 Kafka

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **Amazon MSK**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **Amazon MSK**. Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this Amazon MSK Destination definition.

Brokers: List of Kafka brokers to connect to. (E.g., `kafkaBrokerHost: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.

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`.

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 for filtering and grouping at the final destination. 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 Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Clear Persistent Queue** button. A maximum queue size of 1 GB disk space is automatically allocated per Worker Process. If the queue fills up, Cribl Edge will block outbound data.

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, queueing is stopped and data blocking is applied. Enter a numeral with units of KB, MB, etc.

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.

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 button if you want to flush out files that are currently queued for delivery to this Destination. A confirmation modal will appear. (Appears only after **Output ID** has been defined.)

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 **Advanced Settings** > **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: Reject certificates that are **not** authorized by a CA in the **CA certificate path**, or by another trusted CA (e.g., 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.

Maximum TLS version: Optionally, select the maximum TLS version to use when connecting.

Authentication

Use the **Authentication Method** buttons to select an AWS authentication method.

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 Edge Workers 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 Workers not in an AWS VPC, e.g., 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

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.

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, i.e., 10 seconds. Valid range is `1000` to `3600000` ms, i.e., 1 second to 1 hour.

Request timeout (ms): Maximum time to wait for Kafka to respond to a request. Defaults to `60000` ms, i.e., 1 minute.

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.

Authentication timeout (ms): Maximum time to wait for Kafka to respond to an authentication request.

Reauthentication threshold (ms): Specifies a time window during which Cribl Edge can reauthenticate if needed. Creates the window measuring backwards from the moment when credentials are set to expire.

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:

- `__topicOut`
- `__key`
- `__headers`
- `__keySchemaIdOut`
- `__valueSchemaIdOut`

10.6. Metrics

10.6.1. Graphite

Cribl Edge supports sending data to a Graphite backend Destination.

> Type: Streaming | TLS Support: No | PQ Support: Yes

Configuring Cribl Edge to Output to a Graphite Backend

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **Metrics** > **Graphite**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **Metrics** > **Graphite**.
Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this Graphite definition.

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`.

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 for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

> This section is displayed only when **General Settings** > **Destination protocol** is set to `TCP`, and only when **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 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.

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 button if you want to flush out files that are currently queued for delivery to this Destination. A confirmation modal will appear. (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.

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`.

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.

10.6.2. StatsD

Cribl Edge supports sending data to a StatsD Destination.

> Type: Streaming | TLS Support: No | PQ Support: Yes

Configuring Cribl Edge to Output via StatsD

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **Metrics** > **StatsD**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **Metrics** > **StatsD**.
Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this StatsD definition.

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`.

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 for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

> This section is displayed only when **General Settings** > **Destination protocol** is set to `TCP`, and only when **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 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.

Queue fallback 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** drops the newest events from being sent out of Cribl Edge, and throws away incoming data, while leaving the contents of the PQ unchanged.

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`.

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.

10.6.3. StatsD Extended

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

Configuring Cribl Edge to Output via StatsD Extended

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **Metrics** > **StatsD Extended**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **Metrics** > **StatsD Extended**. Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this StatsD Extended definition.

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`.

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 for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

> This section is displayed only when **General Settings** > **Destination protocol** is set to `TCP`, and only when **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 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.

Queue fallback 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** drops the newest events from being sent out of Cribl Edge, and throws away incoming data, while leaving the contents of the PQ unchanged.

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`.

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.

10.7. New Relic Ingest

10.7.1. New Relic Events

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

Configuring Cribl Edge to Output Events to New Relic

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **New Relic Ingest** > **Events**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **New Relic Ingest** > **Events**. Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this Destination.

Region: Select which New Relic region endpoint to use.

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.)

Authentication Settings

> 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.

Authentication method: Select one of the following buttons.

- **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.

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 for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

> This section 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 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.

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 button if you want to flush out files that are currently queued for delivery to this Destination. A confirmation modal will appear. (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 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.

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.

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 before blocking. This is set per Worker Process. Defaults to `5`.

Max body size (KB): Maximum size of the request body. Defaults to `1000` KB.

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.

10.7.2. New Relic Logs & Metrics

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.

Configuring Cribl Edge to Output to New Relic

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **New Relic Ingest** > **Logs & Metrics**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **New Relic Ingest** > **Logs & Metrics**. Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this New Relic definition.

Region: Select which New Relic region endpoint to use.

Authentication Settings

> 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.

Authentication method: Select one of the following buttons.

- **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.

Optional Settings

Log type: Name of the `logType` to send with events. E.g., `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.

- **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.)

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 for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

This section 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 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.

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 button if you want to flush out files that are currently queued for delivery to this Destination. A confirmation modal will appear. (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.

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.

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. Defaults to `1000` KB.

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.

10.8. Prometheus

10.8.1. Prometheus

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

Configuring Cribl Edge to Output to Prometheus

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **Prometheus**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **Prometheus**. Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this Prometheus output definition.

Remote Write URL: The endpoint to send events to, e.g.: `http://localhost:9200/write`

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 for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

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 is displayed when the **Backpressure behavior** is set to **Persistent Queue**.
>
> On Cribl-managed Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Clear Persistent Queue** button. A maximum queue size of 1 GB disk space is automatically allocated per Worker Process. If the queue fills up, Cribl Edge will block outbound data.

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, queueing is stopped and data blocking is applied. Enter a numeral with units of KB, MB, etc.

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: Click this button if you want to flush out files that are currently queued for delivery to this Destination. A confirmation modal will appear. (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_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.

Advanced Settings

Validate server certs: Reject certificates that are **not** authorized by a trusted CA (e.g., 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. Defaults to `4096` KB.

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 2 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.

10.8.2. Grafana Cloud

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

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, click **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, click **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.

Configuring Cribl Edge to Output to Grafana Cloud

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **Grafana Cloud**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **Grafana Cloud**.
Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this Grafana Cloud output definition.

Loki URL: The endpoint to send log events to, e.g.: `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, e.g.: `https://prometheus-blocks-prod-us-central1.grafana.net/api/prom/push` . This is the **Remote Write Endpoint** URL you wrote down earlier.

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 for filtering and grouping at the final destination. 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 Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Clear Persistent Queue** button. A maximum queue size of 1 GB disk space is automatically allocated per Worker Process. If the queue fills up, Cribl Edge will block outbound data.

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, queueing is stopped and data blocking is applied. Enter a numeral with units of KB, MB, etc.

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: Click this button if you want to flush out files that are currently queued for delivery to this Destination. A confirmation modal will appear. (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.

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. E.g.: `<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.

Advanced Settings

Validate server certs: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (e.g., 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. Defaults to `4096` KB.

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 slider 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 2 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.

10.8.3. Loki

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

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **Loki**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **Loki**. Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this Loki output definition.

Loki URL: The endpoint to send events to, e.g.: `https://logs-prod-us-central1.grafana.net`.

> To proxy outbound HTTP/S requests, see System Proxy Configuration.

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 for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

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 is displayed when the **Backpressure behavior** is set to **Persistent Queue**.
>
> On Cribl-managed Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Clear Persistent Queue** button. A maximum queue size of 1 GB disk space is automatically allocated per Worker Process. If the queue fills up, Cribl Edge will block outbound data.

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, queueing is stopped and data blocking is applied. Enter a numeral with units of KB, MB, etc.

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: Click this button if you want to flush out files that are currently queued for delivery to this Destination. A confirmation modal will appear. (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

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.

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 (e.g., 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. Defaults to `4096` KB.

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 slider 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.

10.9. Splunk

10.9.1. Splunk HEC

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
>
> 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.

Configuring Cribl Edge to Output to Splunk HEC Destinations

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **Splunk** > **HEC**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **Splunk** > **HEC**. Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this Splunk HEC definition.

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: URL of a Splunk HEC endpoint to send events to (e.g., `http://myhost.example.com:8088/services/collector/event`). This setting appears only when **Load balancing** is toggled to `No`.

Splunk HEC Endpoints

The **Splunk HEC Endpoints** table is where you specify a known set of receivers on which to load-balance data. It appears only when **Load balancing** is toggled to `Yes`.

Click **Add Endpoint** to specify more receivers on new rows. Each row provides the following fields:

HEC Endpoint: Specify the URL to a Splunk HEC endpoint to send events to – e.g., `http://localhost:8088/services/collector/event`.

Load weight: Set each connection's relative traffic-handling capability 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.

> For Splunk Cloud endpoints, change the **Splunk HEC endpoint**'s default `http:` prefix to: `https:`.

Authentication Settings

Use the **Authentication method** buttons to select one of these options:

- **Manual**: Displays an **HEC Auth token** field for you to enter your Splunk HEC API access token.
- **Secret**: This option exposes an **HEC 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.

This Destination does not support Mutual TLS (mTLS).

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.splunkcloud.com/services/collector/event
response: {"text":"Data channel is missing","code":10}
time: 2023-02-14T15:26:03.413Z
```

Optional Settings

Exclude current host IPs: This slider 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 for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

This section 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 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.

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 button if you want to flush out files that are currently queued for delivery to this Destination. A confirmation modal will appear. (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.

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 (e.g., 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 before blocking. This is set per Worker Process. Defaults to `5`. 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.

The next two fields 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.

> 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.

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 2 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 Resources

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, click 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.

10.9.2. Splunk Single Instance

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.

For a **Standard** Splunk Cloud instance whose `../default/outputs.conf` file contains multiple indexer entries, you must instead use Cribl Edge's Splunk Load Balanced Destination.

> Type: Streaming | TLS Support: Configurable | PQ Support: Yes

Configuring Cribl Edge to Output to Splunk Destinations

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles or the **Destinations** left nav, select **Splunk** > **Single Instance**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles, select **Splunk** > **Single Instance**. Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this Splunk Single Instance definition.

Address: Hostname of the Splunk receiver.

Port: The port number on the host.

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 for filtering and grouping at the final destination. 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 Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Clear Persistent Queue** button. A maximum queue size of 1 GB disk space is automatically allocated per Worker Process. If the queue fills up, Cribl Edge will block outbound data.

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.

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** drops the newest events from being sent out of Cribl Edge, and throws away incoming data, while leaving the contents of the PQ unchanged.

Clear persistent queue: Click this button if you want to flush out files that are currently queued for delivery to this Destination. A confirmation modal will appear. (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`:

Validate server certs: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (e.g., 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.

> ### 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 `v3`; `v4` is also available.

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, e.g., `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.

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.

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 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 Resources

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, click 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.

10.9.3. Splunk Load Balanced

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
>
> For additional details about sending to Splunk Cloud, see Splunk Cloud and BYOL Integrations.

Configuring Cribl Edge to Load-Balance to Multiple Splunk Destinations

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **Splunk** > **Load Balanced**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **Splunk** > **Load Balanced**. Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this Splunk LB Destination definition.

Toggling **Indexer discovery** to `Yes` enables automatic discovery of 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 from which 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`. (Worker Nodes/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 indexer list from cluster manager. Defaults to `300` seconds, i.e., 5 minutes.

Authentication method: Use the buttons to select one of these options for authenticating to cluster Manager for indexer discovery:

- **Manual**: In the resulting **Auth token** field, enter the required token.
- **Secret**: This option exposes a **Auth token (text secret)** drop-down, in which you can select a stored secret that references the auth token. A **Create** link is available to store a new, reusable secret.

> Each Worker Process performs its own indexer discovery according to the above settings.

Destinations

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.

Click **Add Destination** to specify more receivers on new rows. Each row provides the following fields:

- **Address**: Hostname of the Splunk receiver. Optionally, you can paste in a comma-separated list, in `<host>:<port>` format.
- **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 connection host (if not an IP). Otherwise, uses the global TLS settings.
- **Load weight**: Set each connection's relative traffic-handling capability 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 **Destinations** table.

> For details on configuring all load balancing settings, see About Load Balancing.

Optional Settings

Toggle **Exclude current host IPs** to `Yes` 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 for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

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 is displayed when the **Backpressure behavior** is set to **Persistent Queue**.
>
> On Cribl-managed Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Clear Persistent Queue** button. A maximum queue size of 1 GB disk space is automatically allocated per Worker Process. If the queue fills up, Cribl Edge will block outbound data.

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.

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** drops the newest events being sent out of Cribl Edge, throws away incoming data, and leaves the contents of the PQ unchanged.

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**, nor by another trusted CA (e.g., 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.

> **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 this slider 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 `v3`; `v4` 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): 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 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.

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, etc., are also allowed. E.g., `42 MB`. Default value of `0` 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.)

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.

SSL Configuration for Splunk Cloud – Special Note

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-
splunk-cloud_cacert.pem -connect inputs1.<your-splunk-cloud>.splunkcloud.com:9997
```

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 SSL Certificate Configuration.

Step 3. Extract the private key from the Splunk Cloud certificate. At the prompt, you will need the `sslPassword` value from the `outputs.conf`. Using Elliptic Curve keys:

```
openssl ec -in 100_<your-splunk-cloud>_splunkcloud/default/<your-splunk-
cloud>_server_cert.pem -out private.pem
```

If you are using RSA keys, instead use:

```
openssl rsa -in 100_<your-splunk-cloud>_splunkcloud/default/<your-splunk-
cloud>_server_cert.pem -out private.pem
```

Step 4. Extract the public key for the Server Certificate:

```
openssl x509 -in 100_<your-splunk-cloud>_splunkcloud/default/<your-splunk-
cloud>_server_cert.pem -out server.pem
```

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 the Splunk Load Balanced Destination's **General Settings** > **Destinations** section. E.g., 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, and paste it into the Splunk Load Balanced Destination's **Advanced Settings** tab > **Throttling** setting.

- If you enable TLS including cert validation, indexer discovery might trigger errors. This is because by default, Cribl will get the indexers' IPs from their certs, not their fully qualified domain names (FQDNs).

  As a workaround, use `server.conf` on each indexer, setting `register_forwarder_address = <your.idx.fqdn>`. Cribl will now get that value, and the certs 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 Resources

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, click 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.

10.10. CrowdStrike Falcon LogScale

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, Cribl recommends following LogScale's Cluster Setup – place a load balancer in front of cluster nodes.

We recommend sending events with the `sourceType` field set to a LogScale parser. This tells LogScale which parser to use to extract fields (e.g., `"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.

Configuring Cribl Edge to Output to CrowdStrike Falcon LogScale Destinations

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **CrowdStrike Falcon LogScale**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **CrowdStrike Falcon LogScale**.

Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this CrowdStrike Falcon LogScale definition.

LogScale endpoint: URL of a CrowdStrike Falcon LogScale endpoint to send events to (e.g., `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.

Authentication Settings

Use the **Authentication method** buttons to select one of these options:

- **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.

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 for filtering and grouping at the final destination. 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**.

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.

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 button if you want to flush out files that are currently queued for delivery to this Destination. A confirmation modal will appear. (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.

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 (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. (This setting is available only when **General Settings** > **Load balancing** is set to `No`.)

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 before blocking. This is set per Worker Process. Defaults to `5`. 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 2 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.

10.11. Datadog

Cribl Edge can send log and metric events to Datadog. (Datadog supports metrics only of type `gauge`, `counter`, and `rate` via its REST API.)

Cribl Edge sends events to the following Datadog endpoints in the US region. Use a DNS lookup to discover and include the corresponding IP addresses in your firewall rules' allowlist.

- Logs: `https://http-intake.logs.datadoghq.com/v1/input`
- Metrics: `https://api.datadoghq.com/api/v1/series`

> Type: Streaming | TLS Support: Yes | PQ Support: Yes

Configuring Cribl Edge to Output to Datadog

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **Datadog**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **Datadog**. Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this Destination definition.

Authentication Settings

Use the **Authentication method** buttons to select one of these options:

- **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.

Optional Settings

Datadog site: Select the Datadog region you are sending to. Defaults to `US`; the other options are `US3`, `US5`, `Europe`, `US1-FED`, and `AP1`.

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 (i.e., 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 (e.g., `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 and UX.

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 for filtering and grouping at the final destination. 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 Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Clear Persistent Queue** button. A maximum queue size of 1 GB disk space is automatically allocated per Worker Process. If the queue fills up, Cribl Edge will block outbound data.

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`. Select `Gzip` to enable compression.

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 button if you want to flush out files that are currently queued for delivery to this Destination. A confirmation modal will appear. (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.

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.

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. Defaults to `4096` KB.

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

10.12. DataSet

Cribl Edge can send log events to DataSet. This Destination sends batches of events, as JSON, to the DataSet API's addEvent method.

> Type: Streaming | TLS Support: Yes | PQ Support: Yes

Configuring Cribl Edge to Output to DataSet

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **DataSet**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **DataSet**. Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this Destination definition.

Authentication Settings

Use the **Authentication method** buttons to select one of these options:

- **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`.

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. If 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> }`.

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>`.

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. 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.

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 for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

> This section 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 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`. Select `Gzip` to enable compression.

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 button if you want to flush out files that are currently queued for delivery to this Destination. A confirmation modal will appear. (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.

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 (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.

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. Defaults to `4096` KB.

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.

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.

10.13. Elasticsearch

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

Configuring Cribl Edge to Output to Elasticsearch

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **Elasticsearch**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **Elasticsearch**. Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this Elasticsearch Destination definition.

Load balancing: When enabled (default), lets you specify multiple bulk API 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**.

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 (e.g., `http://<myElasticCluster>:9200/_bulk`). For an Elastic Cloud deployment, enter its Cloud ID. This setting is not available when **Load balancing** is enabled.

Bulk API URLs

The **Bulk API URLs** table is where you specify a known set of receivers on which to load-balance data. Click **Add URL** to specify more receivers on new rows. Each row provides the following fields:

URL: Specify the URL to an Elastic node to send events to – e.g., `http://elastic:9200/_bulk`

Load weight: Set each connection's relative traffic-handling capability 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.

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.

Authentication Settings

Authentication enabled: When toggled to `Yes`, use the **Authentication method** buttons to 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.

Optional Settings

Exclude current host IPs: This slider 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 for filtering and grouping at the final destination. 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 Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Clear Persistent Queue** button. A maximum queue size of 1 GB disk space is automatically allocated per Worker Process. If the queue fills up, Cribl Edge will block outbound data.

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.

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 button if you want to flush out files that are currently queued for delivery to this Destination. A confirmation modal will appear. (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.

Advanced Settings

Validate server certs: Reject certificates that are **not** authorized by a trusted CA (e.g., 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 before blocking. This is set per Worker Process. Defaults to `5`.

Max body size (KB): Maximum size of the request body. Defaults to `4096` KB.

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). 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`.

> 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).

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`

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 2 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.

10.14. Filesystem/NFS

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

Configuring Cribl Edge to Output to Filesystem Destinations

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **Filesystem**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **Filesystem**. Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this Filesystem definition.

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.

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. (This 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. File name prefix and suffix expressions do not bypass this behavior.
>
> 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 for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

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:

- 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.

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.

Row group size: Set the target memory size for row group segments. Modify this value to optimize memory use when writing. Value must be a positive integer smaller than the **File size** value, with appropriate units. Defaults to `16 MB`.

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`.

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 `V1`; `V2` is also available, but not all reader implementations support it.

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.

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.

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 slider'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 to `Yes` 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).

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.

10.15. Honeycomb

Cribl Edge supports sending events to a Honeycomb dataset.

> Type: Streaming | TLS Support: Yes | PQ Support: Yes

Configuring Cribl Edge to Output to Honeycomb

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **Honeycomb**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **Honeycomb**. Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this Honeycomb definition.

Dataset name: Name of the dataset to send events to. (E.g., `iLoveObservabilityDataset`.)

Authentication Settings

Use the **Authentication method** buttons to select one of these options:

- **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.

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 for filtering and grouping at the final destination. 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 Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Clear Persistent Queue** button. A maximum queue size of 1 GB disk space is automatically allocated per Worker Process. If the queue fills up, Cribl Edge will block outbound data.

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.

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 button if you want to flush out files that are currently queued for delivery to this Destination. A confirmation modal will appear. (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.

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.

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. Defaults to `4096` KB.

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 2 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.

10.16. InfluxDB

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

Configuring Cribl Edge to Output to InfluxDB

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **InfluxDB**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **InfluxDB**. Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this InfluxDB definition.

Write API URL: The URL of an InfluxDB cluster to send events to. (E.g., `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.)

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. (E.g., `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 for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

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 is displayed when the **Backpressure behavior** is set to **Persistent Queue**.
>
> On Cribl-managed Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Clear Persistent Queue** button. A maximum queue size of 1 GB disk space is automatically allocated per Worker Process. If the queue fills up, Cribl Edge will block outbound data.

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.

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 button if you want to flush out files that are currently queued for delivery to this Destination. A confirmation modal will appear. (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.

Advanced Settings

Validate server certs: Reject certificates that are **not** authorized by a trusted CA (e.g., 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. Defaults to `4096` KB.

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.

10.17. MinIO

MinIO is a non-streaming Destination type, to which Cribl Edge can output objects.

> Type: Non-Streaming | TLS Support: Configurable | PQ Support: No

Configuring Cribl Edge to Output to MinIO Destinations

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **MinIO**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **MinIO**. Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this MinIO definition.

MinIO endpoint: MinIO service URL (e.g., 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. E.g., 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. (This field is not displayed

or available on Cribl.Cloud-managed Edge Nodes.)

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.

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.

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**.

> 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 for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

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** buttons to select one of these options:

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 Edge Workers 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 Workers not in an AWS VPC, e.g., 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:

- 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.

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.

Row group size: Set the target memory size for row group segments. Modify this value to optimize memory use when writing. Value must be a positive integer smaller than the **File size** value, with appropriate units. Defaults to `16 MB`.

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`.

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`.

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.

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. Minimun part size is 5 MB. A value of `1` will upload the file's contents sequentially, with a maximum concurrency of `1`. Defaults to `4`.

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 slider'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 to `Yes` 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).

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 (e.g., 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

See also AWS Sources/Destinations & S3-Compatible Stores for information on common errors.

10.18. OpenTelemetry (OTel)

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 and Metric 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:

- For traces, opentelemetry-proto/trace.proto at v0.9.0 · open-telemetry/opentelemetry-proto
- For metrics, opentelemetry-proto/metrics.proto at v0.9.0 · open-telemetry/opentelemetry-proto

The OTel Destination will drop non-conforming events. Also, when trying to convert an event to OTLP, if the Destination encounters a parsing error, it discards the event, and Cribl Edges logs the error.

Configuring Cribl Edge to Output to OTel

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **OpenTelemetry**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **OpenTelemetry**.
Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this OTel output definition.

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.

Optional Settings

Protocol: Use the drop-down to choose the protocol to use when sending data: `gRPC` (the default), or `HTTP`. When set to `HTTP`, the UI displays two additional settings:

- **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 traces 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.

Backpressure behavior: Whether to block, drop, or queue events when all receivers are exerting backpressure.

Tags: Optionally, add tags that you can use for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

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 (e.g., 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.

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. E.g.: `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.

Persistent Queue Settings

> This tab is displayed when the **Backpressure behavior** is set to **Persistent Queue**.
>
> On Cribl-managed Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Clear Persistent Queue** button. A maximum queue size of 1 GB disk space is automatically allocated

> per Worker Process. If the queue fills up, Cribl Edge will block outbound data.

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, queueing is stopped and data blocking is applied. Enter a numeral with units of KB, MB, etc.

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: Click this button if you want to flush out files that are currently queued for delivery to this Destination. A confirmation modal will appear. (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.

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.

Common Settings

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. Defaults to `4096` KB.

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 (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. 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.

10.19. SignalFx

Cribl Edge supports sending events to SignalFx.

> Type: Streaming | TLS Support: Yes | PQ Support: Yes

Configuring Cribl Edge to Output to SignalFx

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **SignalFx**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **SignalFx**. Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this SignalFx definition.

Realm: SignalFx realm name (e.g., `us0`). Required.

Authentication Settings

Use the **Authentication method** buttons to select one of these options:

- **Manual**: Displays an **Auth token** field for you to enter your SignalFx API access token. See SignalFx's Manage Tokens topic.
- **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.

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 for filtering and grouping at the final destination. 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 Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Clear Persistent Queue** button. A maximum queue size of 1 GB disk space is automatically allocated per Worker Process. If the queue fills up, Cribl Edge will block outbound data.

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.

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 button if you want to flush out files that are currently queued for delivery to this Destination. A confirmation modal will appear. (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.

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.

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. Defaults to `4096` KB.

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

For details on integrating with SignalFx, see the SignalFx Developers Guide. Especially relevant is the SignalFx HTTP Send Metrics Reference.

10.20. SNMP Trap

Cribl Edge supports forwarding of SNMP Traps out.

> Type: Streaming | TLS Support: No | PQ Support: No

Configuring Cribl Edge to Forward to SNMP Traps

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **SNMP Trap**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **SNMP Trap**. Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this SNMP Trap definition.

SNMP Trap destinations: Click **Add Destination** to specify more SNMP destinations to forward traps to, on new rows. Each row provides the following fields:

- **Address**: Destination host.
- **Port**: Destination port. Defaults to `162`.

Optional Settings

Tags: Optionally, add tags that you can use for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

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

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 (e.g., 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.

10.21. Sumo Logic

Cribl Edge can send log and metric events to Sumo Logic over HTTP.

> Type: Streaming | TLS Support: Configurable | PQ Support: Yes

Configuring Cribl Edge to Output to Sumo Logic

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **Sumo Logic**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **Sumo Logic**. Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this Sumo Logic Destination definition.

API URL: Enter the URL of the Sumo Logic HTTP Source to which events should be sent. (E.g., `https://endpoint6.collection.us2.sumologic.com/receiver/v1/http/<long-hash>`.)

Optional Settings

Custom source name: Optionally, override the source name configured on the Sumo Logic HTTP Source. This value will be sent with events via the `X-Sumo-Name` HTTP header. You can set the `__sourceName` field on events to assign a custom source name at the event level.

Custom source category: Optionally, override the source category configured on the Sumo Logic Collector and/or HTTP Source. This value will be sent with events via the `X-Sumo-Category` HTTP header. You can set the `__sourceCategory` field on events to assign a custom source category at the event level.

Backpressure behavior: Whether to block, drop, or queue events when all receivers are exerting backpressure.

Tags: Optionally, add tags that you can use for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

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.

Persistent Queue Settings

> This tab is displayed when the **Backpressure behavior** is set to **Persistent Queue**.
>
> On Cribl-managed Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Clear Persistent Queue** button. A maximum queue size of 1 GB disk space is automatically allocated per Worker Process. If the queue fills up, Cribl Edge will block outbound data.

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.

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 button if you want to flush out files that are currently queued for delivery to this Destination. A confirmation modal will appear. (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.

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.

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. Defaults to `4096` KB.

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.

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 Sumo Logic as a metric event. Otherwise, Cribl Edge will send it as a log event.

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 2 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.

10.22. Syslog

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 Understanding Syslog Format Options below, to ensure that your configuration will structure compliant outbound events that downstream services can read.

Configuring Cribl Edge to Output Data in Syslog Format

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **Syslog**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **Syslog**. Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this Syslog definition.

Protocol: The network protocol to use for sending out syslog messages. Defaults to `TCP`; `UDP` is also available.

Load balancing: This option is displayed only when the **Protocol** is set to `TCP`. When enabled (default), lets you specify multiple destinations. 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**.

The following two fields appear **only** with the `No` setting.

Address: Address/hostname of the receiver.

Port: Port number to connect to on the host.

Max record size: 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 <= the MTU (maximum transmission unit for the network path to the Destination system).

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 `Syslog`, the 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 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: Throttle rate, in bytes per second. Defaults to `0`, meaning no throttling. Multiple-byte units such as KB, MB, GB etc. are also allowed, e.g., `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: 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 for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Load Balancing Settings

> Load balancing is available only when the **Protocol** is set to `TCP`.

Enabling the **Load balancing** slider replaces the static **General Settings** > **Address** and **Port** fields with the following controls:

Exclude current host IPs: This slider 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

The **Destinations** table is where you specify a known 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 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 each connection's relative traffic-handling capability 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 section is displayed only for **TCP**, and only when **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 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.

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 button if you want to flush out files that are currently queued for delivery to this Destination. A confirmation modal will appear. (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`:

Validate server certs: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (e.g., 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 two options are always displayed:

Octet count framing: When enabled, Cribl Edge prefixes messages with their byte count. When disabled, Cribl Edge omits prefixes, and instead appends a `\n` to messages.

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 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.

Understanding Syslog Format Options

Unlike other Cribl Edge Destinations, Syslog 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 5424 and/or RFC 5425) before it is transmitted to downstream services.

The Syslog Destination's **General Settings** page offers several settings to format the timestamps, to format the message delivering the event, and to set the syslog-specific default settings for Facility, Severity, and App Name.

Below are two examples of RFC-compliant syslog events:

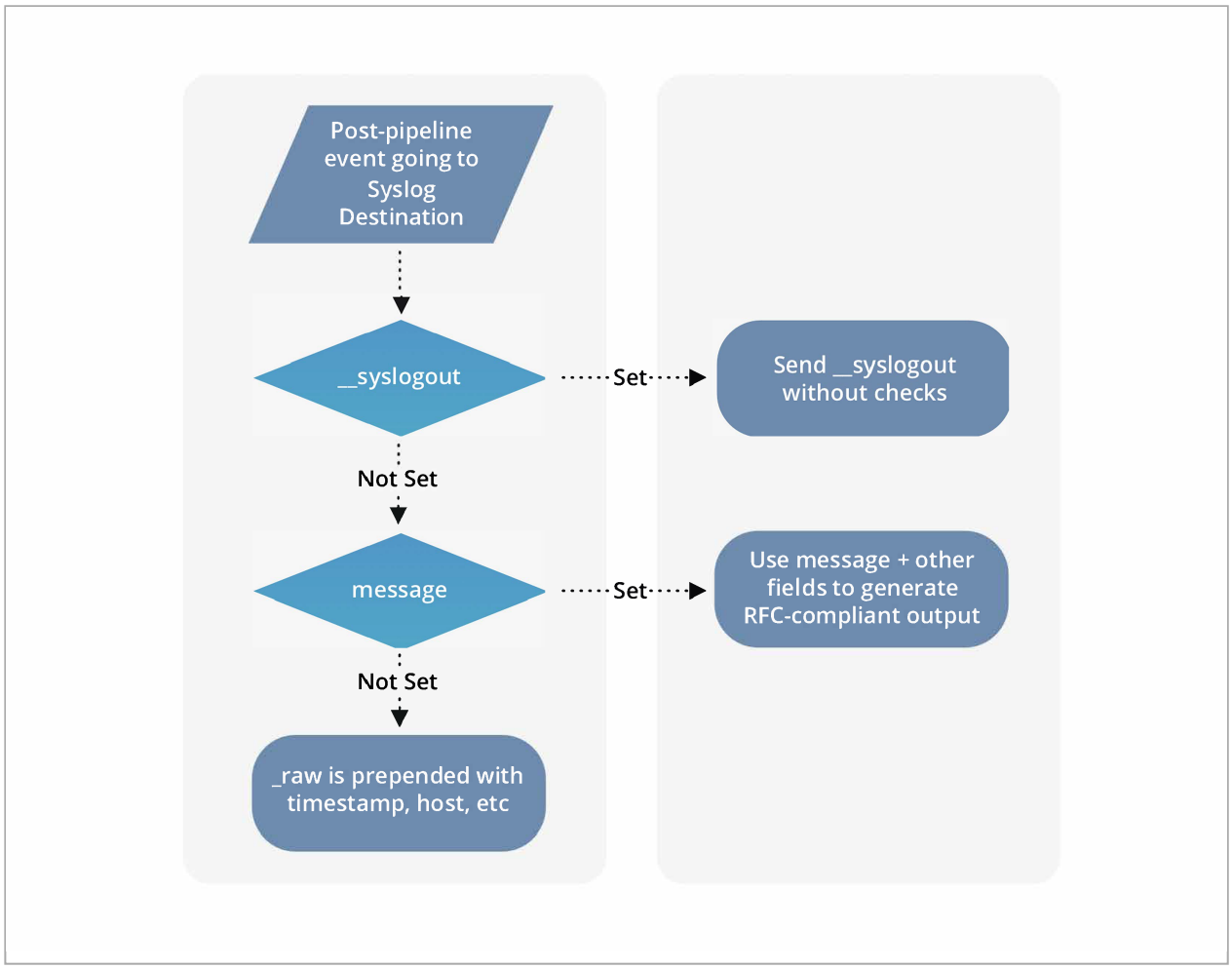
- `<13>Jul 11 10:34:35 testbox testing[42]`
- `<214>1 2022-07-11T18:58:45.000Z testbox testing[42]: foo=bar this=that base=ball gizmo=sprocket`

Cribl Edge offers three different output approaches in the Syslog Destination. The flowchart below summarizes how Cribl Edge determines which approach to use:

- `message` : For ease of use, Cribl recommends using this option. When you define `message`, 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. To use this method, you must configure `message` and must **not** set `__syslogout`.

- `__syslogout` : When you define `__syslogout`, Cribl Edge sends it as the entire syslog message, discards `_raw`, and ignores all the other fields.

- `_raw` (with or without a header): If you didn't define `message` or `__syslogout`, then Cribl Edge uses the existing `_raw` field as the `message` field, and prepends all the other syslog-related fields to the event.



Syslog output flow

The subsections below walk you through some considerations for each of these options, before you configure this Destination. First, let's take a look at internal fields, since they play a critical role in formatting.

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`
- `__host`
- `__syslogout`

Defining `message`

This approach is easiest and least error-prone, because Cribl Edge creates the payload for you. To define the `message`, all you need to supply are the following required fields:

- `__facility`
- `__severity`
- `__host`

Or, to make this even simpler, you can substitute `__priority` for `__facility` and `__severity`. Either way, Cribl Edge will create a new payload using a combination of these required fields. For details on the fields assembled here, see Important Fields below.

Exporting `__syslogout`

As a second approach, if you choose to send `__syslogout` to downstream services, it is exclusive – it becomes the entire syslog message sent. Neither `_raw`, nor any other metadata, will be sent downstream. Also, Cribl Edge will not check to ensure that the value of `__syslogout` is RFC-compliant.

The result will be a proper syslog message only if you hand-build the event yourself. You must manually construct the `__syslogout` payload, starting with `_time`, for all the fields that Cribl Edge would automatically handle with the above `message` option. In particular:

> When defining `__syslogout`, you must follow the syslog protocol's RFCs. Otherwise, downstream services will misinterpret or completely ignore the message. Some syslog receivers might drop non-compliant events entirely, or try to “fix” the format by supplying missing fields.

The most common uses for the `__syslogout` method are:

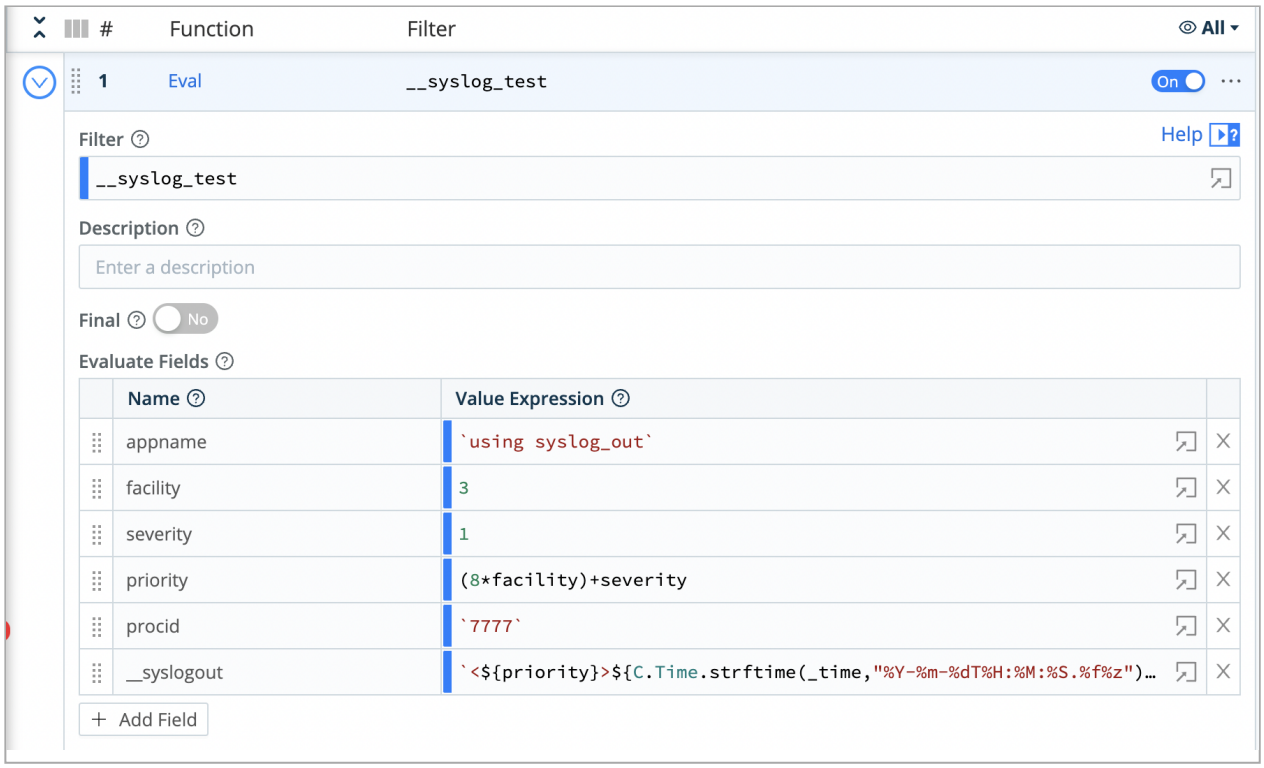
- When the value of `_raw` is already in syslog format as it comes in, and minimal processing is necessary.
- When sending raw TCP data to a destination that doesn't enforce syslog RFCs, such as a raw TCP listener. Cribl Edge currently does not provide a native "raw TCP" Destination. However, in some environments, configuring a Syslog Destination with the TCP protocol and `__syslogout` is an effective method for delivering raw data over TCP.

Constructing `__syslogout`

You'll need to add `_time` to the payload. For example, in an Eval Function, you could use **Evaluate fields** to build up `__syslogout` in the expression below. Here, the `priority` encodes the `severity + facility`, according to the syslog protocol.

NAME	VALUE EXPRESSION
`priority`	`(8*facility)+severity`
`__syslogout`	`` `<${priority}>${C.Time.strftime(_time,"%Y-%m-%dT%H:%M:%S.%f%z")} ${host} ${appname} [${procid}]: ${_raw}` ``

Here's that example expression in a full Function:

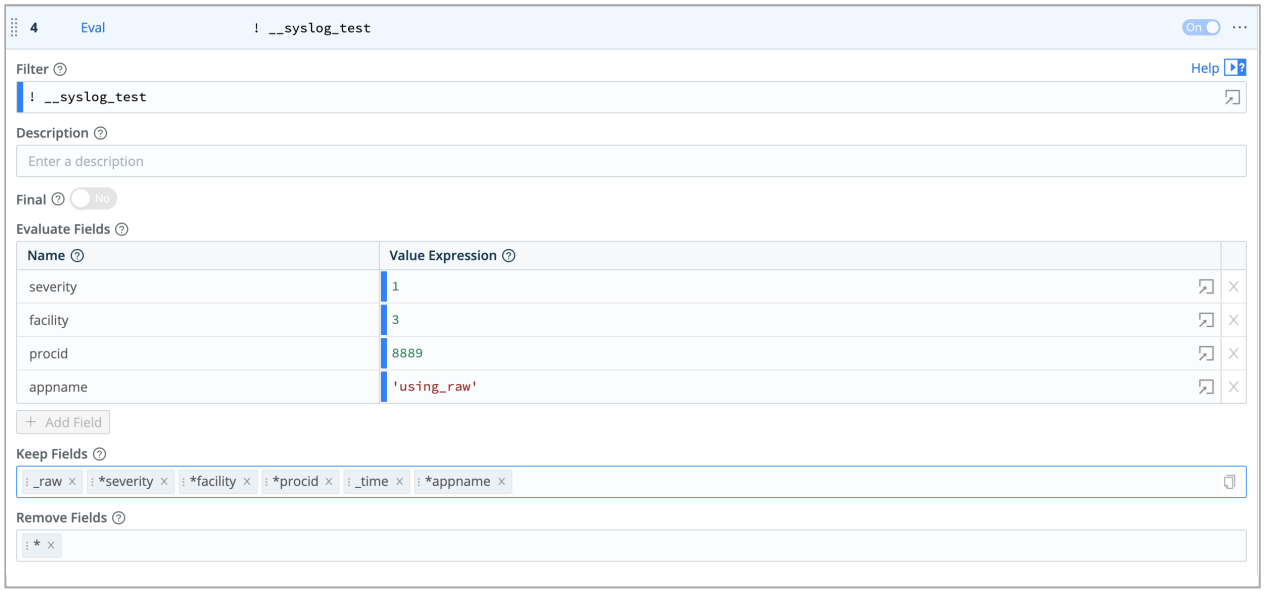


Adding `__syslogout`, `_time` to construct a valid syslog message

Enhancing `_raw` with syslog

A third approach is to add the message content in `_raw`, and construct the syslog "envelope" around `_raw` by including the `severity`, `priority`, `facility`, `procid`, `msgid`, and `appname` fields, as required.

Here's an alternative Eval Function that illustrates this:



Adding syslog decorations to `_raw`

Both Eval Functions are provided in this example Pipeline:

syslog_loop.json

```
{
  "id": "syslog_loop",
  "conf": {
    "output": "default",
    "groups": {},
    "asyncFuncTimeout": 1000,
    "functions": [
      {
        "filter": "true",
        "conf": {
          "mode": "reserialize",
          "type": "json",
          "srcField": "_raw",
          "dstField": "_raw",
          "keep": [
            "resource",
            "path",
            "httpMethod"
          ],
          "remove": []
        },
        "id": "serde",
        "disabled": false
      },
      {
        "filter": "true",
        "conf": {
          "clones": [
            {
              "__syslog_test": "true"
            }
          ]
        },
        "id": "clone",
        "disabled": false
      },
      {
        "filter": "__syslog_test",
        "conf": {
          "add": [
            {
              "name": "appname",
              "value": "'using_syslogout'"
            },
            {
              "name": "severity",
              "value": "1"
            },
            {
              "name": "facility",
              "value": "3"
            },
            {
              "name": "pri",
              "value": "(8 * facility) + severity"
            },
            {
              "name": "procid",
              "value": "'7777'"
```

```
            },
            {
              "name": "__syslogout",
              "value": "`<${pri}>${C.Time.strftime(_time,\"%b %d %H:%M:%S\")}
${host} ${appname}[${procid}]: ${_raw}`"
            }
          ],
          "keep": [],
          "remove": []
        },
        "id": "eval",
        "disabled": false
      },
      {
        "filter": "! __syslog_test",
        "conf": {
          "add": [
            {
              "name": "severity",
              "value": "1"
            },
            {
              "name": "facility",
              "value": "3"
            },
            {
              "name": "procid",
              "value": "8889"
            },
            {
              "name": "appname",
              "value": "'using_raw'"
            }
          ],
          "keep": [
            "_raw",
            "*severity",
            "*facility",
            "*procid",
            "_time",
            "*appname"
          ],
          "remove": [
            "*"
          ]
        },
        "id": "eval",
        "disabled": false
      }
    ]
  }
}
```

In this method, Cribl Edge takes the value of `_raw` verbatim, and then prepends a human-readable timestamp, host, and other information.

If you are using `_raw` as the `message` field, make sure you explicitly set the `priority` and `facility` whenever possible. Otherwise, Cribl Edge will use the default values. The acceptable values are defined in the RFCs.

> When using `_raw`, you might end up with duplicate fields in the event. For example, even if your event already contains a `timestamp`, Cribl Edge might prepend another `timestamp` to the beginning of `_raw`, without checking whether the data is already present. This can increase event sizes with redundant data. If you use this method, make sure you first strip the prepended (duplicate) fields.
>
> Also, when using `_raw`, this Destination does not check whether there's a valid header defined in the event – it always adds one. Also, because this Destination reformats the data, you might not see the header information when you preview it in Live Capture.

For details on the fields assembled here, see Important Fields below.

Defined `message` and `_raw` Fields

If Cribl Edge's Destination stage receives an event that contains both `message` and `_raw` fields, it will build the syslog package using the `message` field, discarding `_raw`.

The `message` (or `_raw`) field must be an ASCII string in order to be put on the syslog wire. This Destination does not handle JSON objects. Avoid mismatching these types, or else no data might be sent out.

If your intermediate processing – such as a `JSON.parse(_raw)` transformation – has converted the `_raw` field's contents into a JSON object literal, you will need to stringify the result, using a method like `JSON.stringify(_raw)`. You can do so in a post-processing Pipeline attached to the Syslog Destination.

Important Fields

The `Message` and `_raw` prepend methods use additional fields to create the final payload. When using `__syslogout`, Cribl Edge ignores these fields.

The fields below appear in the order generated by a syslog-formatted event.

Sample syslog-formatted event: `<38>1 2022-07-11T22:04:46.000Z testbox app01 [4321] AC-123 - foo=bar this=that base=ball gizmo=sprocket`

- `__facility` or **facility**: Cribl Edge uses this field to calculate priority. The RFC protocol dictates Facility levels. For details, see Facility.

- `__severity` or **severity**: Cribl Edge also uses this field to calculate priority. The RFC protocol dictates Severity levels. For details, see Severity.

- `__priority`: If you configure this field, Cribl Edge will use it and override the `severity` and `facility` values. The priority displays at the beginning of a syslog event, `<38>` in the example above. If you don't configure this field, then Cribl Edge calculates it using 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`. The `priority` of `<38>` in the example above is `8*4(Facility of auth) + 6 (Severity of info)`.

- `_time`: The value of `_time` in Cribl events is in epoch format, but the syslog RFCs dictate that each event's timestamp is must be in human-readable format. When defining a Syslog Destination, you can have an option to use the ISO8601 (recommended) or the syslog format. ISO8601 defines the method for specifying time zone and year, whereas the older syslog format lacks this information. The example above shows the ISO8601 format, listed after the `priority` (`<13>`) and `version`.

- `__host` OR **host**: the value for the required host field in a syslog event, following the timestamp. `testbox` in the example above.

- `__appname` or **appname**: The required application name. `app01` in the example above. This is typically the name of the daemon or process that is logging any given event.

- `__procid` or **procid**: This optional field is the Process ID. `4321` in the example above. Use a numeric value for this field, optionally this may be surrounded with brackets [ ]. Cribl Edge will automatically adjust the spaces and syntax to ensure RFC-compliant formatting.

- `__msgid` or **msgid**: This optional field is the Message ID. `AC-123` in the example above.

For each pair of attribute names (above), Cribl Edge uses the values as specified here:

1. The event contains both `__<key>` and `<key>`: Cribl Edge uses the value of `__<key>` and ignores `<key>` if also present. For example, if the event contains `__host`, the value of **host** will be ignored if also present.

2. The event contains `<key>`: Cribl Edge uses the value of `<key>`. For example, if **host** is set, the value is applied.

3. The event contains neither `__<key>` nor `<key>`, Cribl Edge uses the default values configured in the Syslog Destination.

10.23. TCP JSON

Cribl Edge supports sending data over TCP in JSON format.

> Type: Streaming | TLS Support: Configurable | PQ Support: Yes

Configuring Cribl Edge to Output Data in TCP JSON Format

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **TCP JSON**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **TCP JSON**. Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this Destination definition.

Load balancing: When enabled (default), lets you specify multiple destinations. 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.

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

Exclude current host IPs: This slider 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 etc. are also allowed, e.g., `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 for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Load Balancing Settings

Enabling the **Load balancing** slider replaces the static **General Settings** > **Address** and **Port** fields with the following controls:

Destinations

The **Destinations** table is where you specify a known 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 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 each connection's relative traffic-handling capability 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 is displayed when the **Backpressure behavior** is set to **Persistent Queue**.
>
> On Cribl-managed Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Clear Persistent Queue** button. A maximum queue size of 1 GB disk space is automatically allocated per Worker Process. If the queue fills up, Cribl Edge will block outbound data.

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.

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 button if you want to flush out files that are currently queued for delivery to this Destination. A confirmation modal will appear. (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)

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 (e.g., 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

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.

Format

TCP JSON events are sent in newline-delimited JSON format, consisting of:

1. A header line. Can be empty, e.g.: `{}`. 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.

10.24. Wavefront

Cribl Edge supports sending events to Wavefront analytics.

> Type: Streaming | TLS Support: Yes | PQ Support: Yes

Configuring Cribl Edge to Output to Wavefront

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **Wavefront**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **Wavefront**. Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this Wavefront definition.

Domain name: WaveFront domain name, e.g., `longboard`. Required.

Authentication Settings

Use the **Authentication method** buttons to select one of these options:

- **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.

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 for filtering and grouping at the final destination. 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 Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Clear Persistent Queue** button. A maximum queue size of 1 GB disk space is automatically allocated per Worker Process. If the queue fills up, Cribl Edge will block outbound data.

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.

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 button if you want to flush out files that are currently queued for delivery to this Destination. A confirmation modal will appear. (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.

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.

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. Defaults to `4096` KB.

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.

10.25. Webhook

Cribl Edge can send log and metric events to webhooks and other generic HTTP endpoints.

> Type: Streaming | TLS Support: Configurable | PQ Support: Yes

Configuring Cribl Edge to Output via HTTP

From the top nav, click **Manage**, then select a **Fleet** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Collect** (Edge only). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **Webhook**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The resulting drawer will provide the options below.

Or, to configure via the Routing UI, click **Data** > **Destinations** (Stream) or **More** > **Destinations** (Edge). From the resulting page's tiles or the **Destinations** left nav, select **Webhook**. Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

Output ID: Enter a unique name to identify this HTTP output definition.

URL: Endpoint URL to send events to. The internal field `__url`, where present in events, will override this value. See Internal Fields below.

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. One of:

- `NDJSON (newline-delimited JSON)`: The default.
- `JSON Array`: Arrays in JSON-parseable format.
- `Custom`: A wrapper object that contains batched events. You define the format for both the wrapper and the individual event. See Custom 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 for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

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 (e.g., `token`); or nested, using a period (e.g., `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, e.g.: `` `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, click **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 – e.g., `id` instead of `` `${id}` `` – will be evaluated as strings.

- **Authentication headers**: Optionally, click **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 – e.g., `id` instead of `` `${id}` `` – will be evaluated as strings.

Persistent Queue Settings

> This tab is displayed when the **Backpressure behavior** is set to **Persistent Queue**.
>
> On Cribl-managed Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the **Clear Persistent Queue** button. A maximum queue size of 1 GB disk space is automatically allocated per Worker Process. If the queue fills up, Cribl Edge will block outbound data.

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.

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 button if you want to flush out files that are currently queued for delivery to this Destination. A confirmation modal will appear. (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.

Advanced Settings

Validate server certs: 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). 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. 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 before blocking. This is set per Worker Process. Defaults to `5`.

Max body size (KB): Maximum size of the request body. 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.

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, i.e., what data you want to make available from the event. **Batch expression** defines the wrapper object for batched events, i.e., 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` – i.e., 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}] }` ``.

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 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/Sumo Logic 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 2 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.

11. Working With Data

11.1. 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:

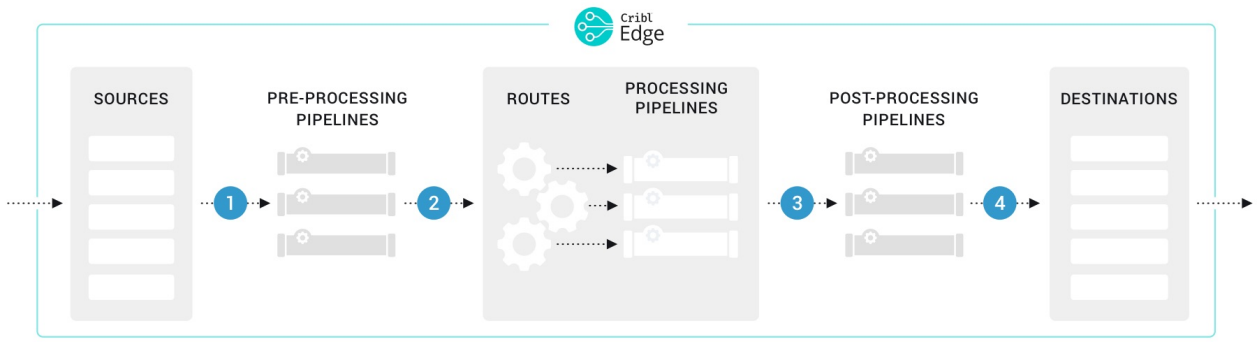
- Fields that start with a double-underscore are known as **internal fields**, and each Source can add one or many to each event. For example, Syslog adds both a `__inputId` and a `__srcIpPort` field. Internal fields are used only within Cribl Edge, and are not passed down to Destinations.
- Upon arrival 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):

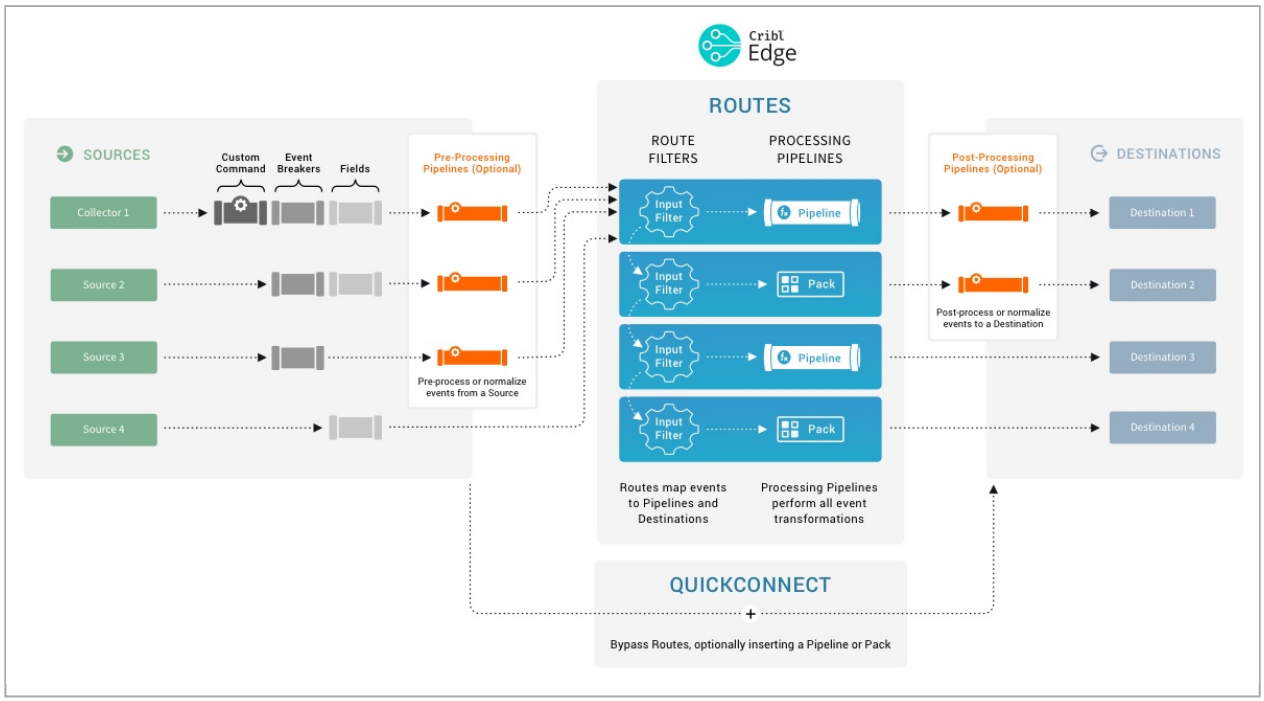
1. Click **Start a Capture**.
2. In the resulting modal, enter a **Filter expression** to narrow down the events of interest.
3. Click **Capture...** and (optionally) change the default Time and/or Event limits.
4. Select the desired **Where to capture** option. There are four options:

- **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.



11.2. 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.

11.3. 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

Select **Routing** > **Data Routes** from Cribl Edge's global top nav (single-instance deployments) or from a Worker Group's/Fleet's top nav (distributed deployments). To configure a new Route, click **Add Route**.

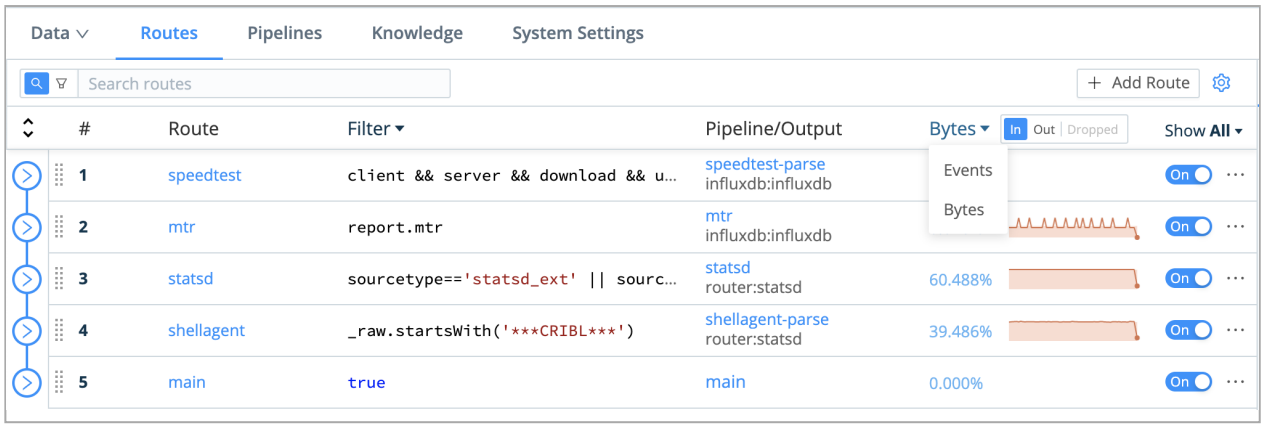
How Do Routes Work

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.



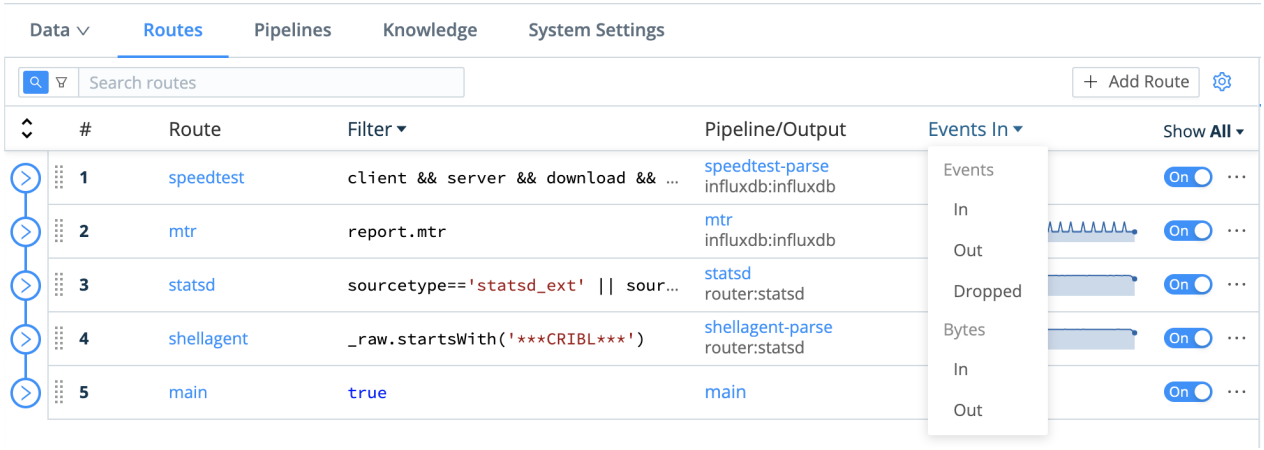
Routes and bytes

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 **main** Route serves as a catch-all for any event that does not match any of the other Routes.

Above, note the selectors 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.

When you condense the Routes page to a narrower viewport, Cribl Edge consolidates the **In**/**Out**/**Dropped** selectors onto an expanded **Bytes**/**Events** drop-down menu, as shown below.



Routes and events (combined menu)

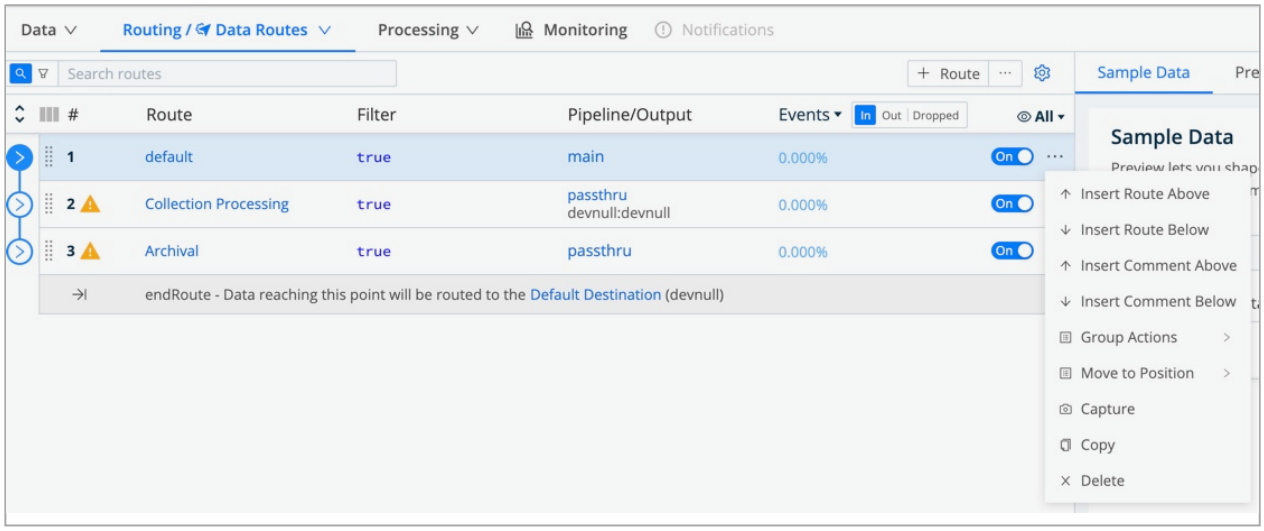
Managing the Routes Page

To apply a Route before another, simply drag it vertically. Use the sliders 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.)

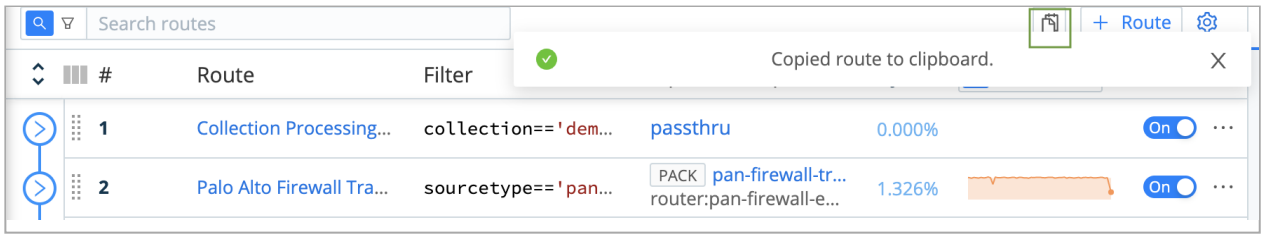
Click a Route's Options (•••) menu to display multiple options:

- Insert, group, move, copy, or delete Routes.
- Insert comments above or below Routes.
- Capture sample data through a selected Route.



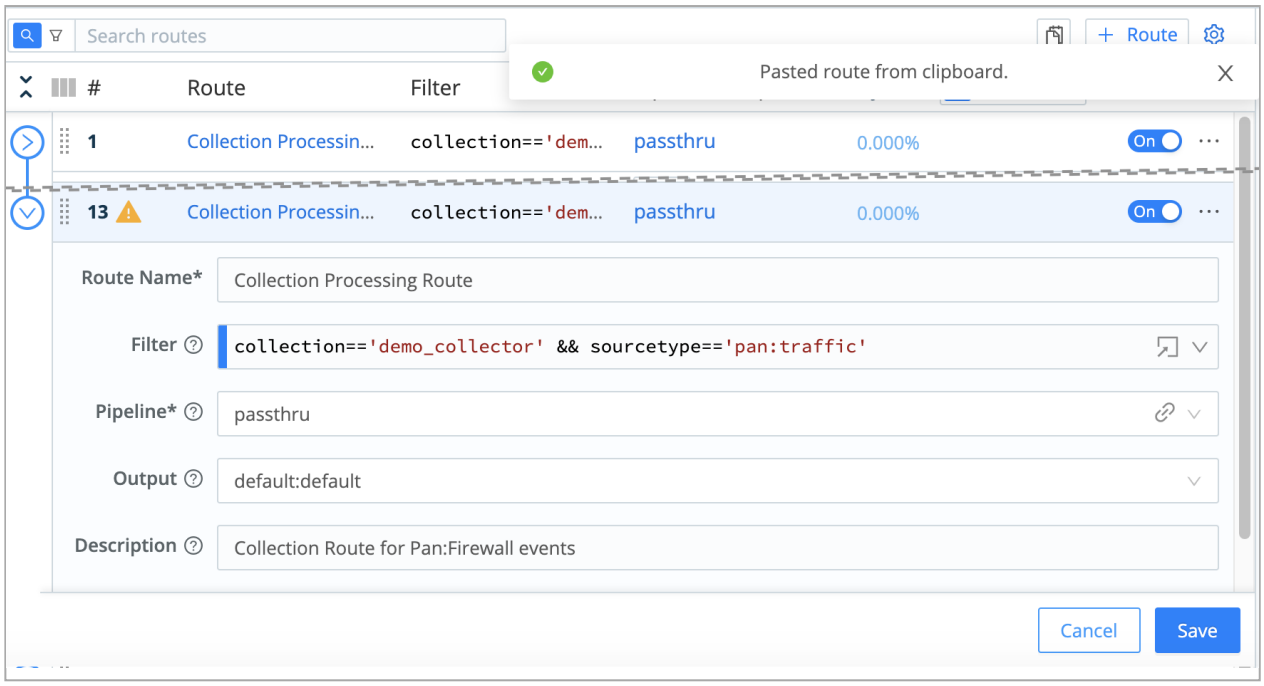
Route > Options menu

Copying a Route displays the confirmation message and the (highlighted) Paste button shown below.



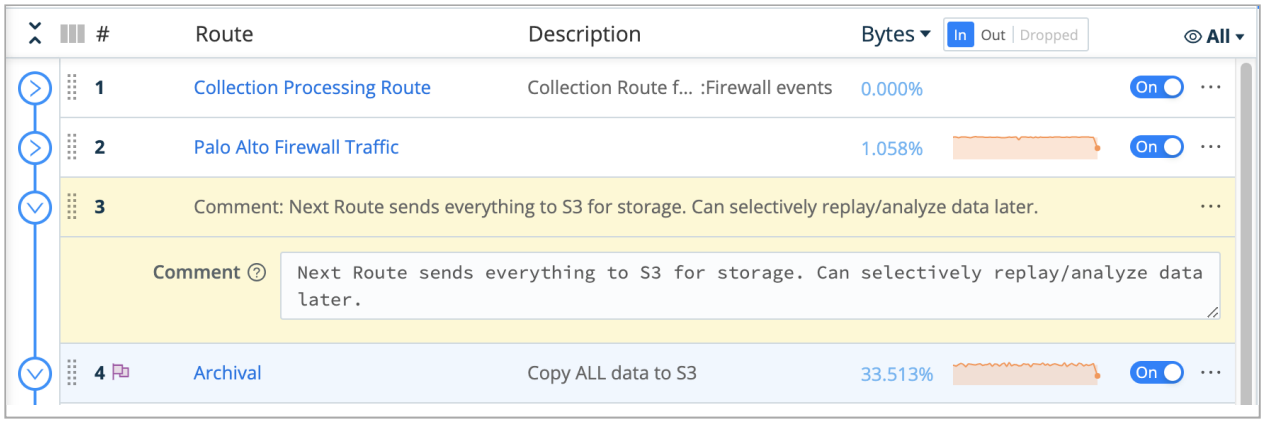
Paste button for copied Route

Pasting creates an exact duplicate of the Route, with a warning indicator to change its duplicate name.



Pasted duplicate Route

Comments work like a Pipeline's Comment Functions: You can use them to describe Routes' purposes and/or interactions.



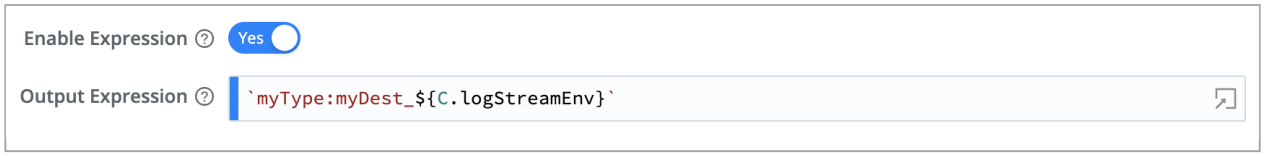
Comments make the Routing table self-documenting

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 the **Enable expression** slider 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: `myType:myDest_${C.logStreamEnv}`. (This evaluation happens at Route construction time, not per event.)



Output expression enabled

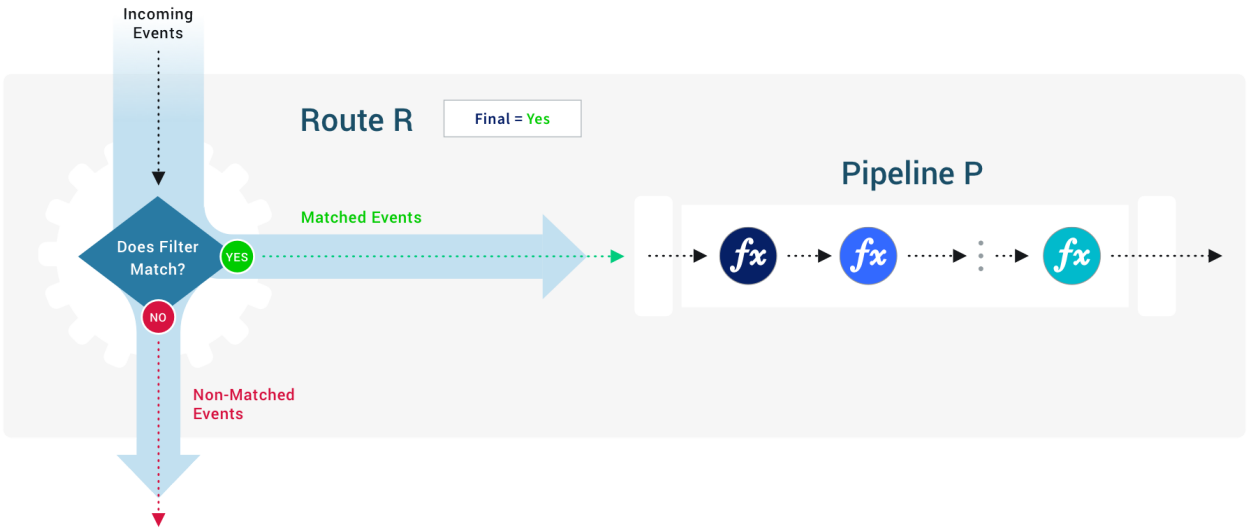
> Expressions that match no Destination name will silently fail.
>
> On Routes within Packs, neither **Output** control is available, because Packs cannot specify Destinations.

The `Final` Toggle

When an event that enters the system and matches a Route-Pipeline pair, it will usually be either:

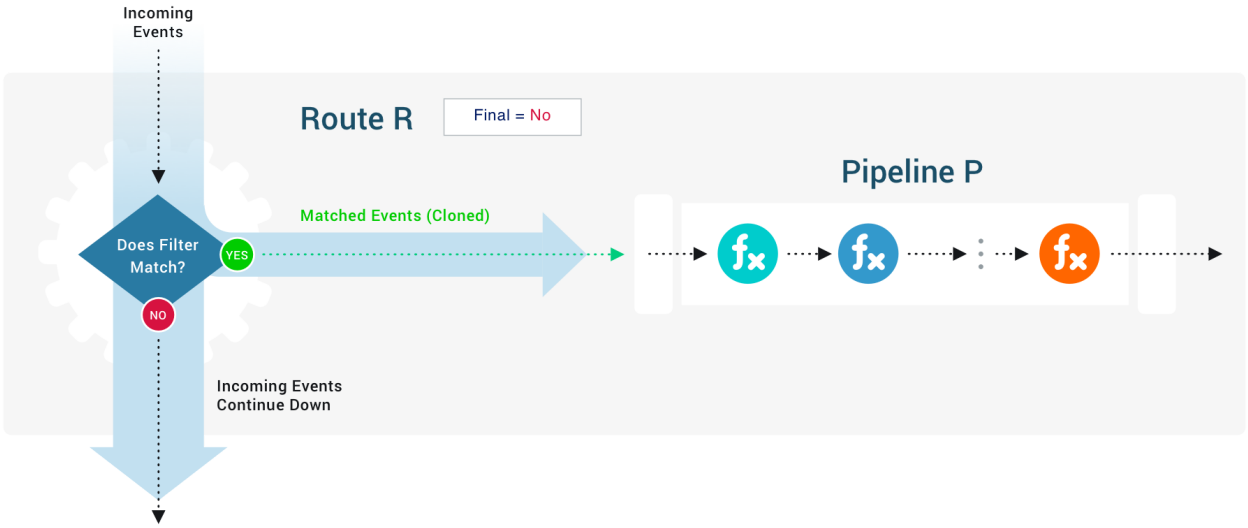
- Dropped by a function, or
- Transformed (optionally) and exit the system.

This behavior is ensured by the `Final` toggle in Route settings. It defaults to `Yes`, meaning that matched events will be **consumed** by that Route, and will not be evaluated against any other Routes that sit below it.



If the `Final` toggle is set to `No`, clone(s) of the matching events will be processed by the configured Pipeline, and the original events will be allowed to continue their trip to be evaluated and/or processed by other

Route-Pipeline pairs. In the diagram below, events will continue down to be evaluated by other Pipelines.



When you set the `Final` toggle to `No`, the **Add Clone** button appears. Click it to open the table shown below. Here, you can add a field – name and value – to the cloned events sent to this Route's Pipeline. Click **Add Field** to define more added fields/values.

Non-final Route: add clone fields

Final Flag and Cloning Considerations

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 cloning is not needed at all (i.e., all `Final` toggles stay at default), then it makes sense to start with the broadest expression at the top, so as to consume as many events as early as possible.
- If cloning is needed on a narrow set of events, then it might make sense to do that upfront, and 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.

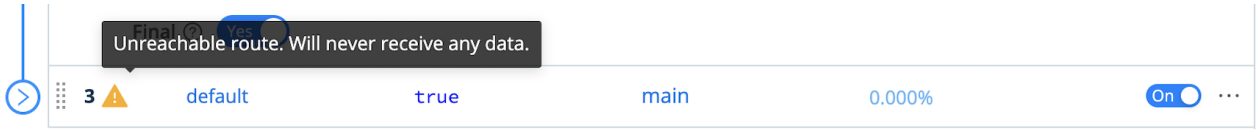
Route Groups

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.

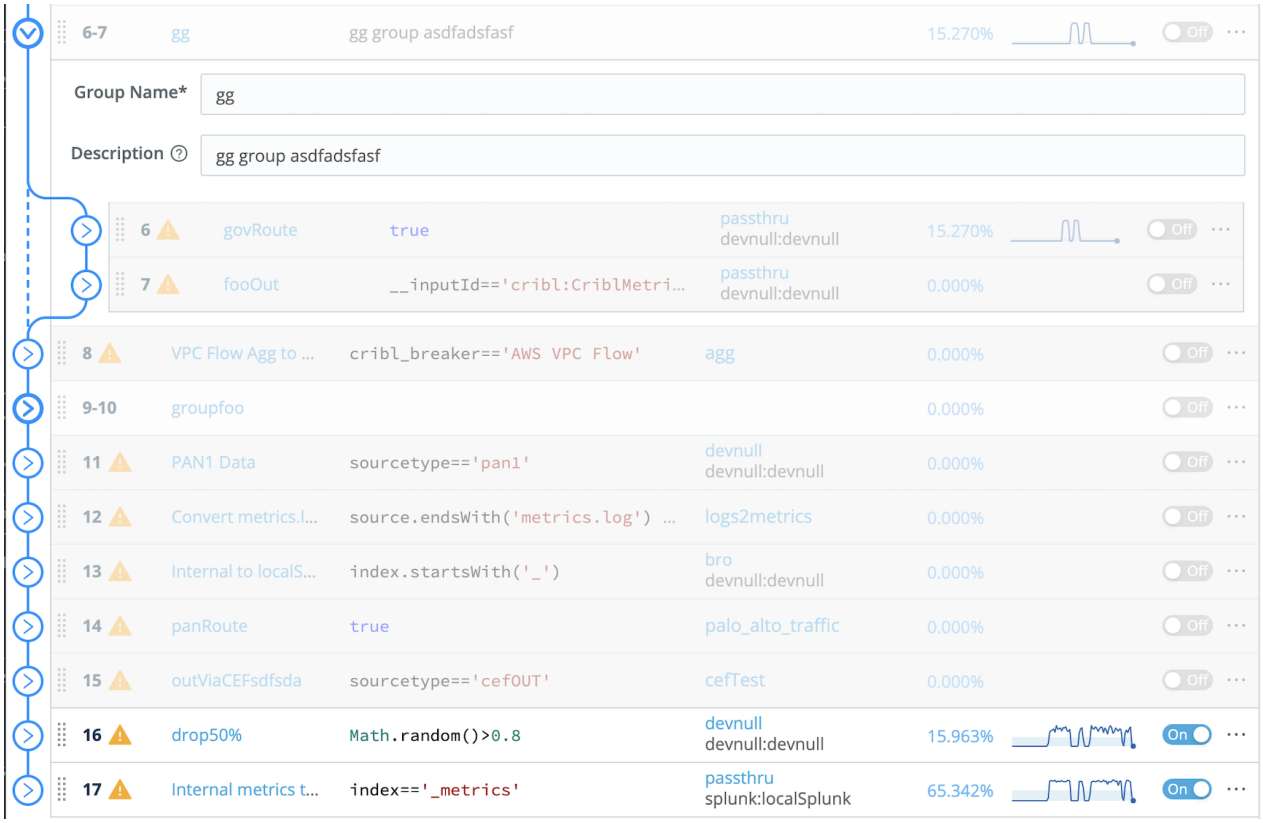


Unreachable Route warning, on hover

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 (slider is set to `On`).
- Previous Route is final (**Final** slider 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).



Unreachable Route warnings, many

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.

11.4. 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.

Accessing Pipelines

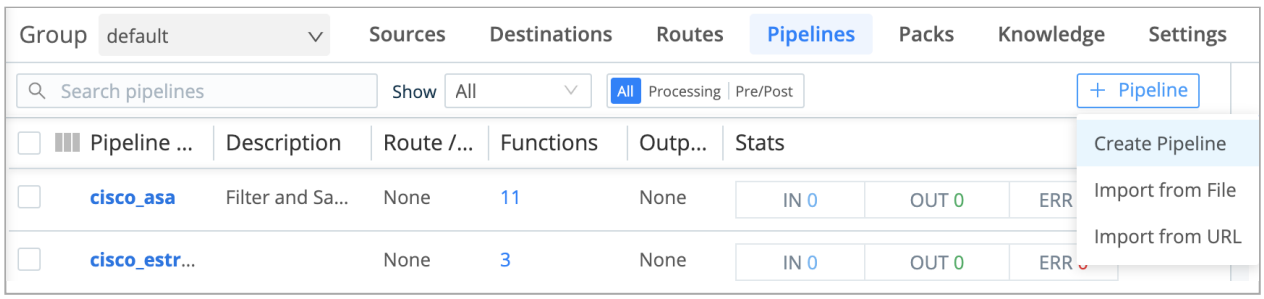
In single-instance deployments, select **More** > **Pipelines**. In distributed deployments, select **Manage** from the top nav, then select a **Fleet** to configure. From a **Fleet's** top nav, select **More** > **Pipelines**.

> After you've clicked into a Pipeline, the right Preview pane adds a Pipeline Status tab, which you can click to see the Pipeline's events throughput.

Adding Pipelines

To create a new Pipeline, or to import an existing Pipeline to a different Cribl Edge instance, click **Add Pipeline** at the upper right. The resulting menu offers three options:

- **Create Pipeline**: Configure a new Pipeline from scratch, by adding Functions in Cribl Edge's graphical UI.
- **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.)

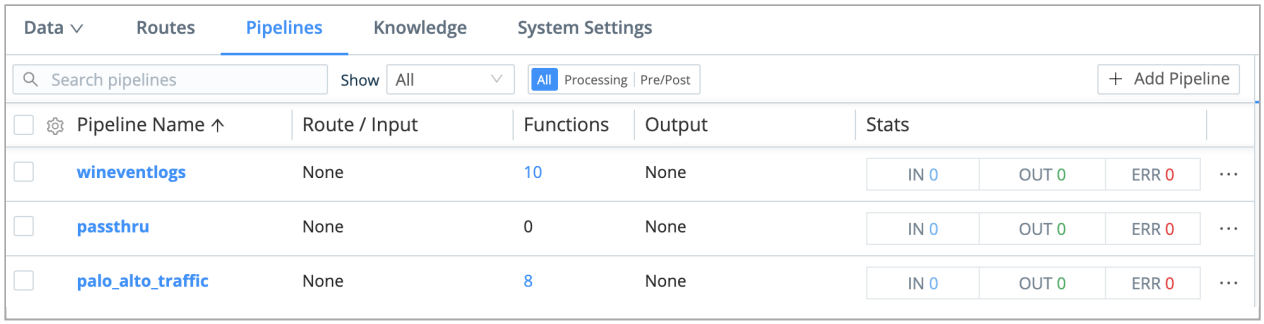


To export a Pipeline, see Advanced Mode (JSON Editor).

To import or export a Pipeline along with broader infrastructure (like Knowledge Objects and/or sample data files), see Packs.

How Do Pipelines Work

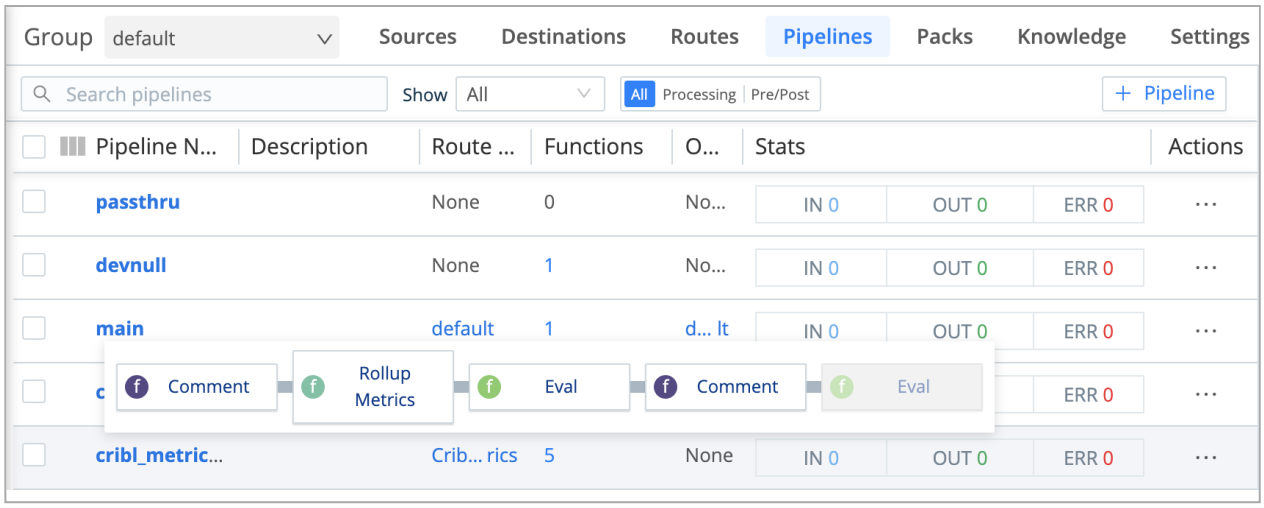
Events are always delivered to the beginning of a Pipeline via a Route. The data in the **Stats** column shown below are for the last 15 minutes.



Pipelines and Route inputs

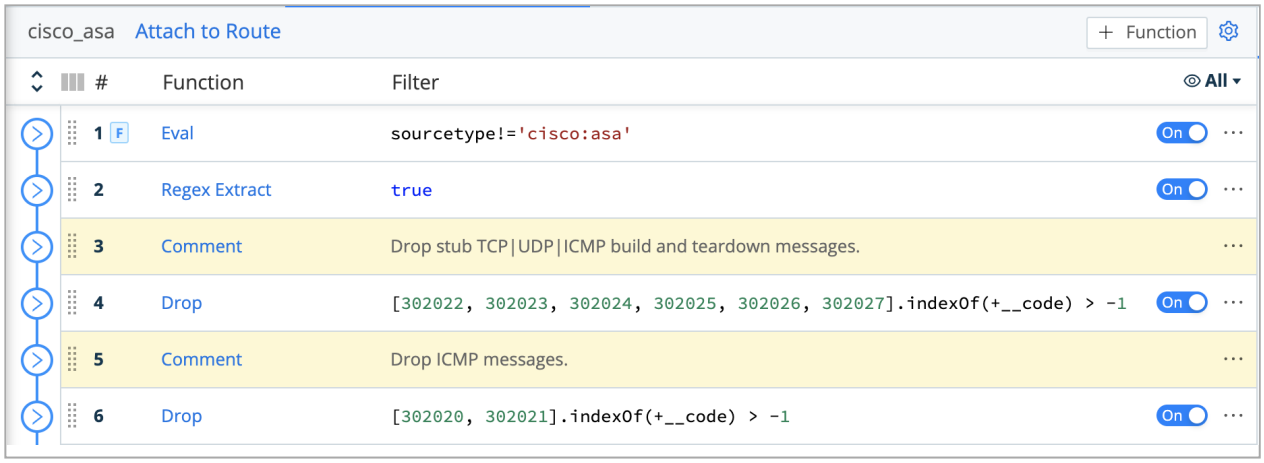
You can press the] (right-bracket) shortcut key to toggle between the Preview pane and an expanded Pipelines display. (This shortcut works when no field has focus.)

In the condensed Pipelines display above, you can also hover over any Pipeline's **Functions** column to see a horizontal preview of the stack of Functions contained in the Pipeline:



Preview on hovering over the bottom 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 upper left to associate a new Pipeline with a Route.
>
> You can streamline a complex Pipeline's display by organizing related Functions into Function groups.

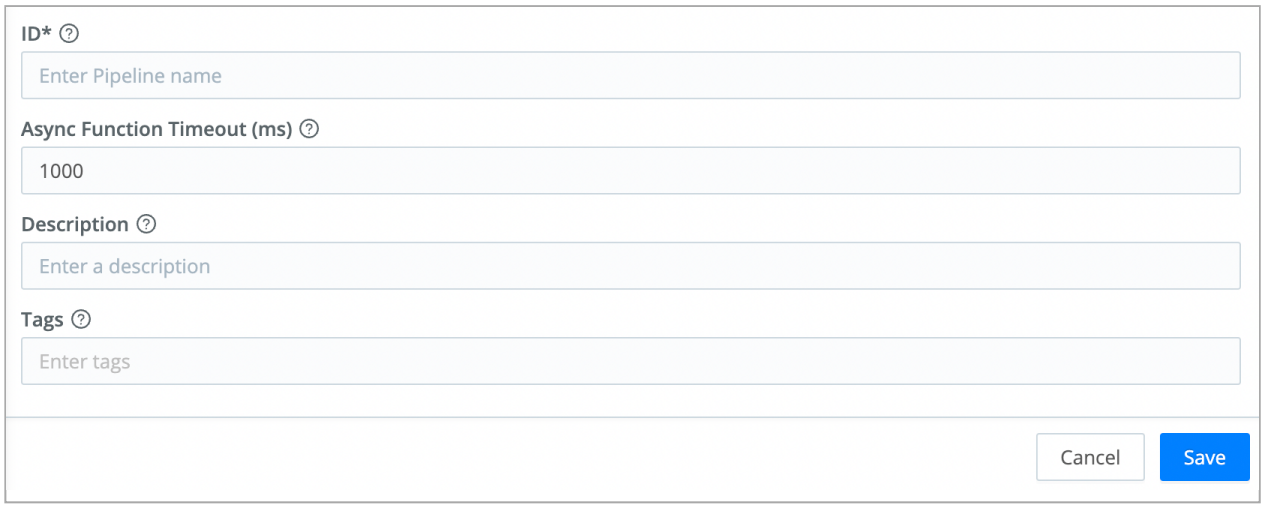
Pipeline Settings

Click 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).

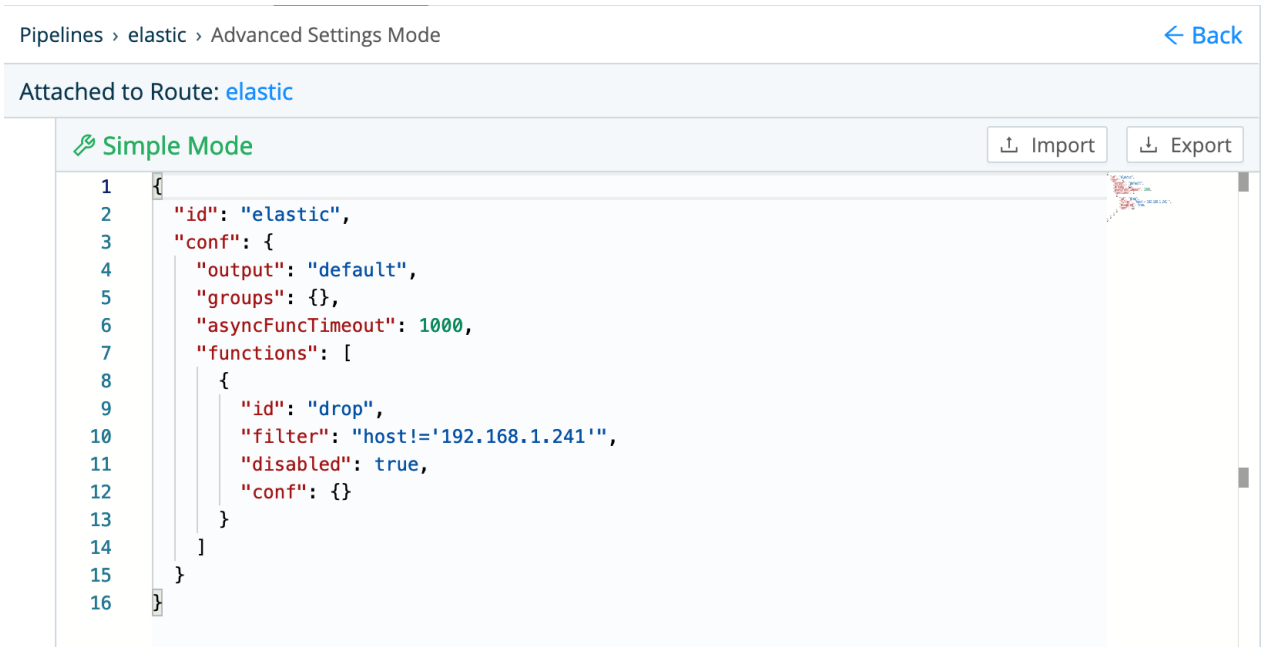
- Use the **Tags** field to attach arbitrary labels to the Pipeline. Once attached, you can use these tags to filter/search and group Pipelines.



Pipeline Settings

Advanced Mode (JSON Editor)

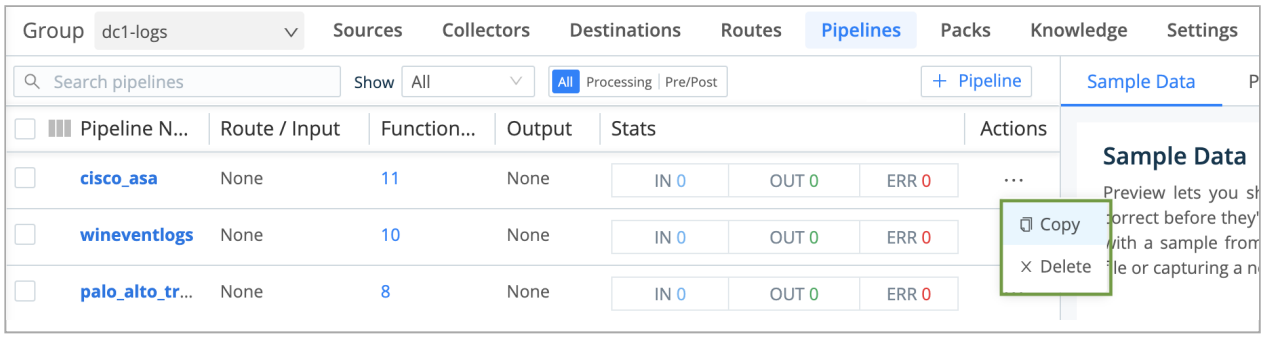
Once you've clicked the gear button to enter Pipeline Settings, you can click **Edit as JSON** at the upper right to edit the Pipeline's definition in a JSON text editor. In this mode's editor, you can directly edit multiple values. You can also use the **Import** and **Export** buttons here to copy and modify existing Pipeline configurations, as `.json` files.



Click **Edit in GUI** at upper right to return to the graphical Pipeline Settings page; then click **Back to <pipeline-name>** to restore the graphical Pipeline editor.

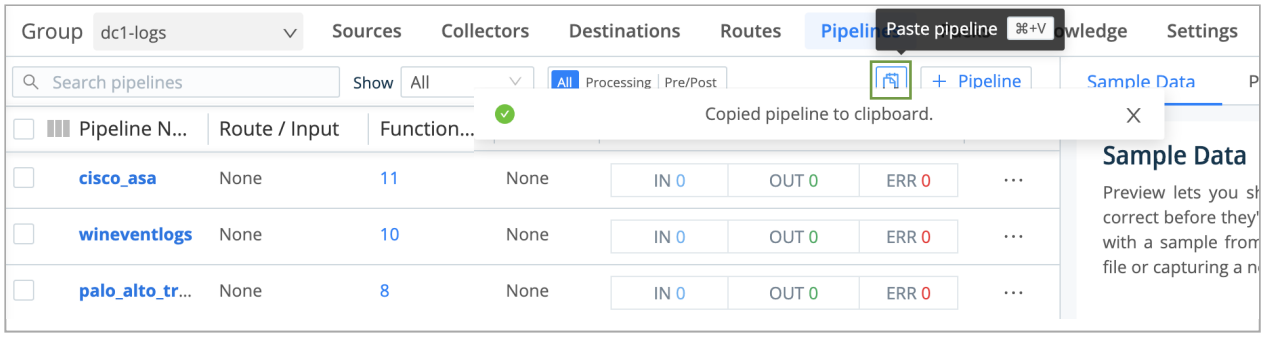
Pipeline Actions

Click a Pipeline's Actions (**...**) menu to display options for copying or deleting the Pipeline.



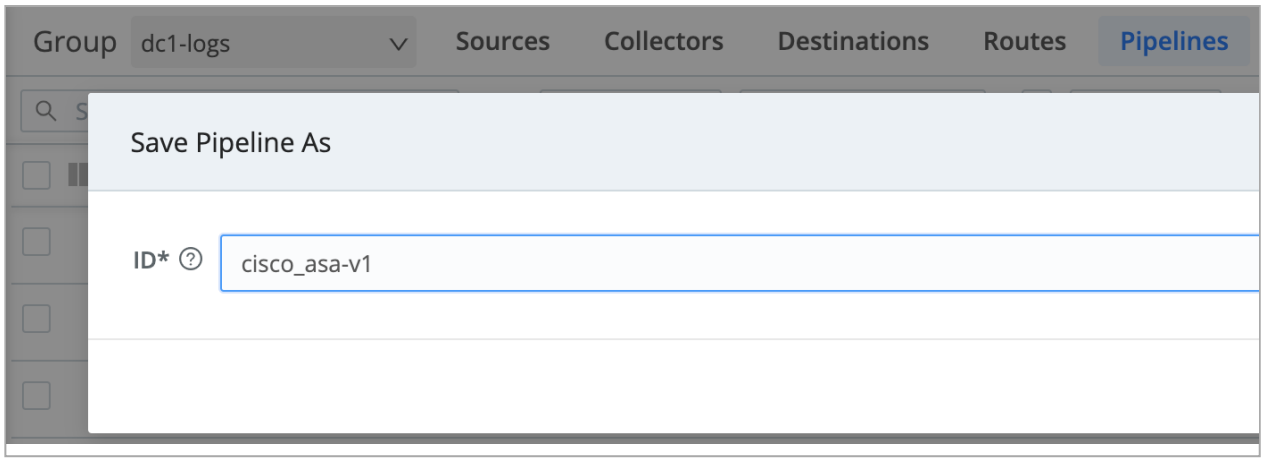
Pipeline > Actions menu

Copying a Pipeline displays the 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.



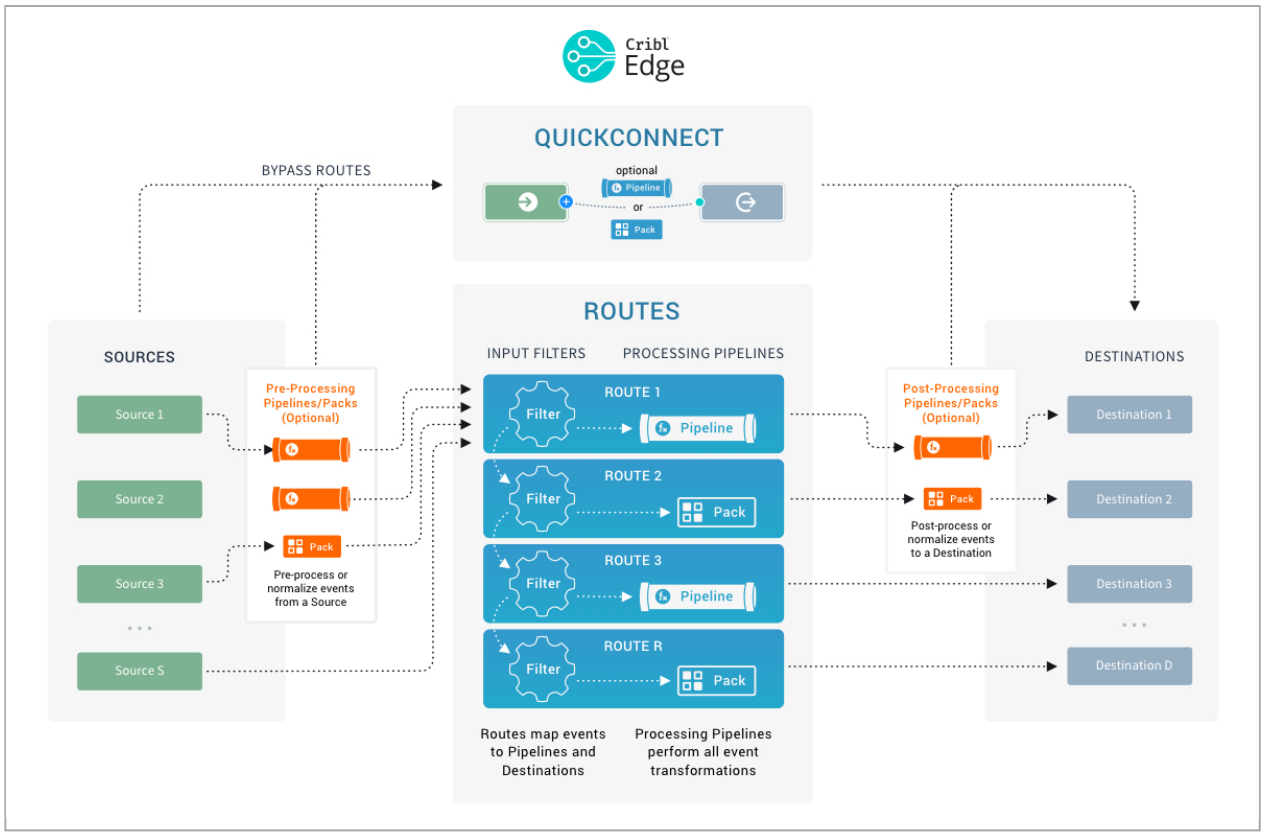
Saving/renaming a pasted Pipeline

Chaining Pipelines

In Cribl Edge 3.2.x and above, you can use the Chain Function to send the output of a Pipeline to another Pipeline or Pack. There are scope restrictions within Packs, and general guardrails against circular references.

Types of Pipelines

You can apply various Pipeline types at different stages of data flow. All Pipelines have the same basic internal structure (a series of Functions) – the types below differ only in their position in the system.



Pre-processing, processing, and post-processing Pipelines

Pre-Processing Pipelines

These are Pipelines that are attached to a Source to condition (normalize) the events **before** they're delivered to a processing Pipeline. They're optional.

Typical use cases are event formatting, or applying Functions to **all** events of an input. (E.g., to extract a `message` field before pushing events to various processing Pipelines.)

You configure these Pipelines just like any other Pipeline, by selecting **Pipelines** from the top menu. 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. (E.g., to ensure that a `_time` field exists for all events bound to a Splunk receiver.)

You configure these Pipelines as normal, by selecting **Pipelines** from the top menu. 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.

11.5. 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 accelerate working with Cribl Support, because they facilitate quickly replicating your Cribl Edge environment.

For example, where a Pipeline's 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. You can develop and test a configuration, and then port it from development to production instances, or readily deploy it to multiple Worker Groups/Fleets.

We don't claim to have brokered world peace here, but we do modestly hope to promote a stable, prosperous Pax Criblatica for the Cribl Edge ecosystem.

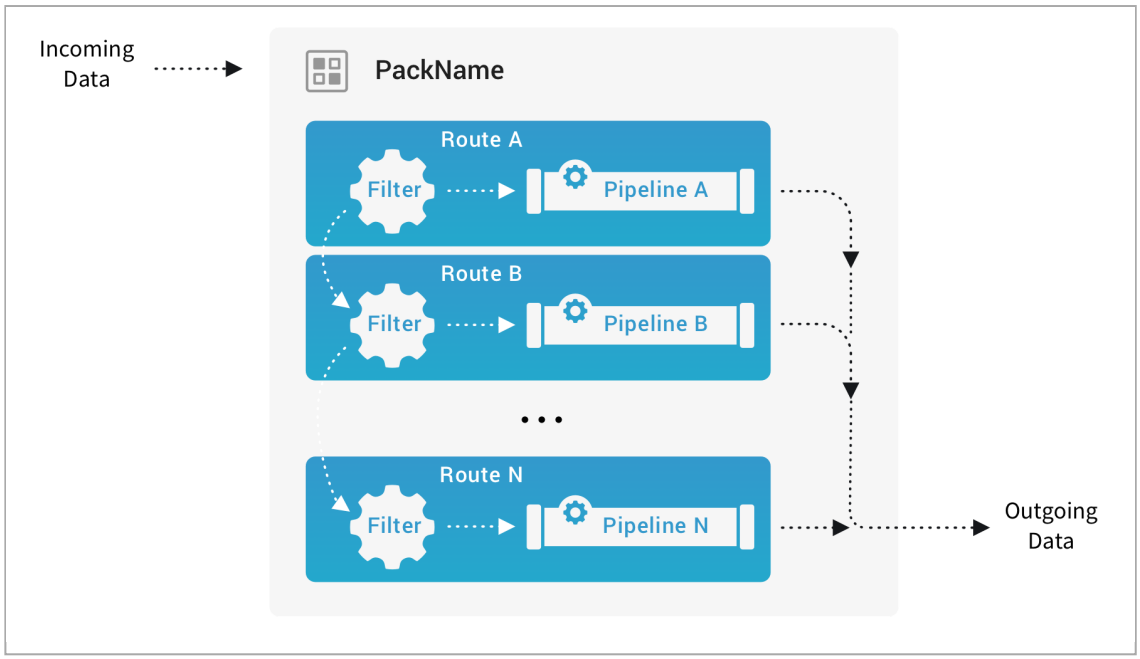
What Is a Pack?

Packs are implemented as a user interface (described on this page) and as a `.crbl` file 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, and Schemas)



A Pack with internal Routes & Pipelines; no Knowledge or samples

As the above list suggests, a Pack can encapsulate a whole set of infrastructure for a given use case.

> Packs don't have access to Global Variables defined outside the Pack. Global Variables defined **within** a Pack are not available to Cribl Edge resources outside the Pack. In either scenario, you must explicitly duplicate the Variables 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 Packs' contained Pipelines.)

You connect a Pack with a Source and Destination by attaching it to a Route (see below), just as you'd attach a Pipeline.

Where Can I Get Some Packs?

Easy now. See The Cribl Packs Dispensary™ below.

Using Packs

These instructions cover using predefined Packs, as well as creating and modifying Pack configurations.

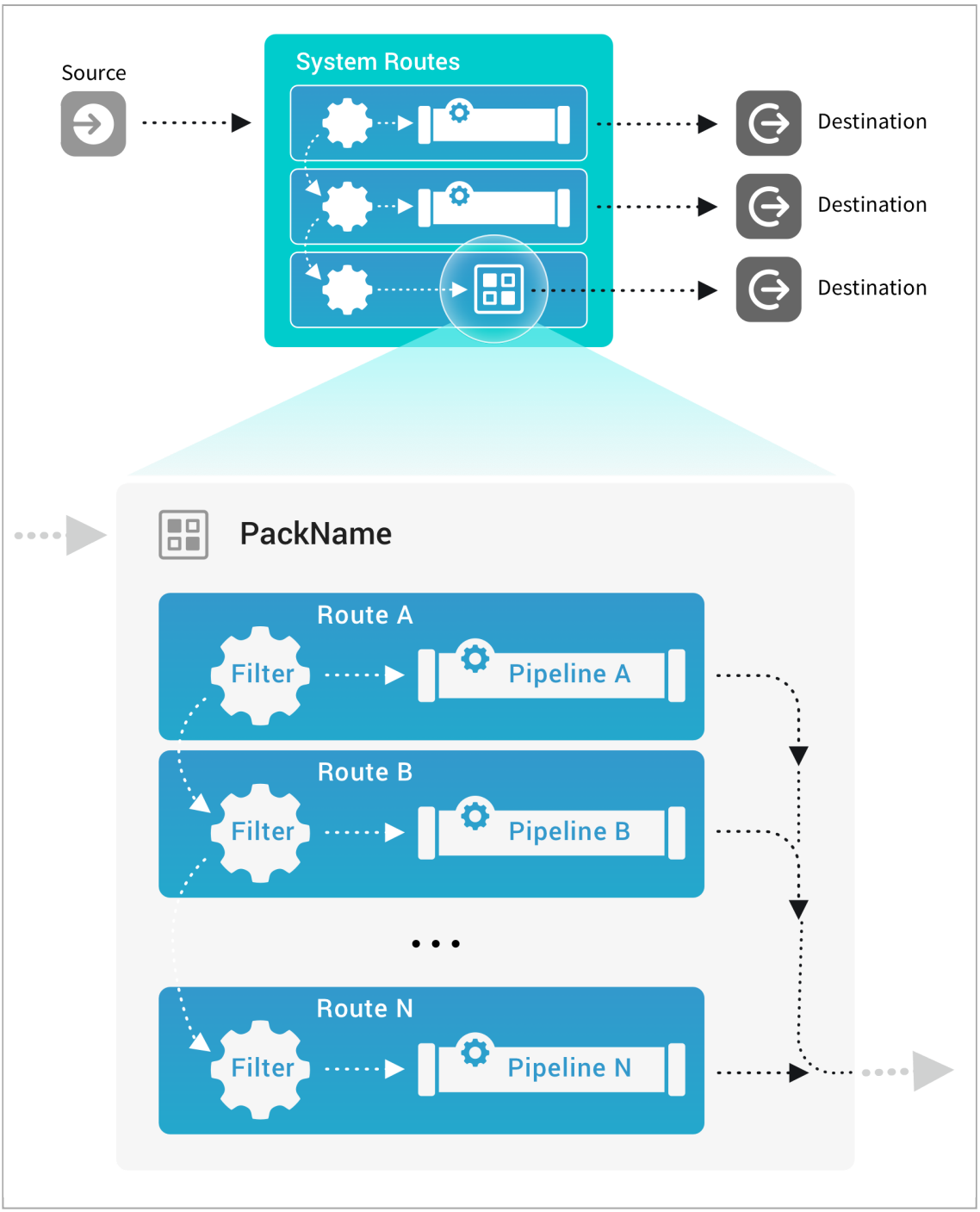
> **Version Compatibility**
>
> Packs created or modified in Cribl Edge 4.0.x cannot be used in any pre-4.0 version of Cribl Edge. (If you try, you'll see a `should NOT have additional properties` error.) To avoid this problem, Cribl recommends that you upgrade to Cribl Edge 4.0.x or later.
>
> For compatibility questions about any individual Packs, please contact us in Cribl Community Slack's `#packs` channel.

Where Can I Use Packs?

Wherever you can reference a Pipeline, you can specify a Pack:

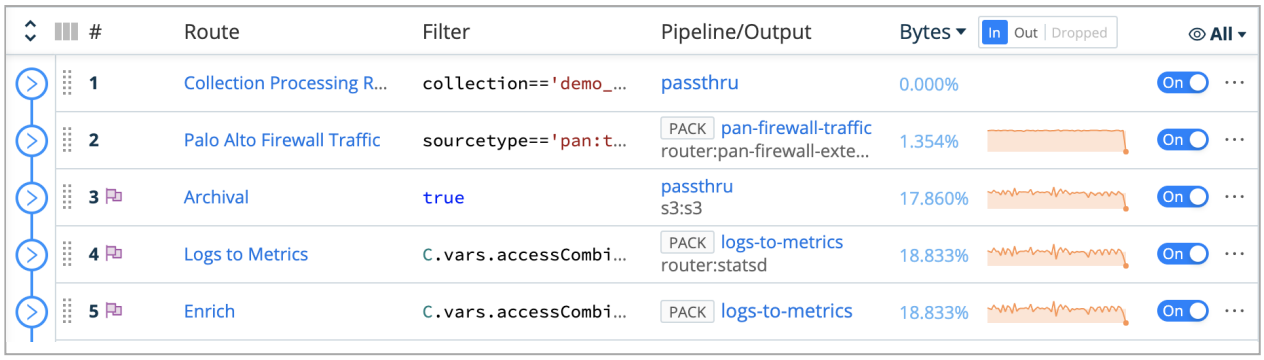
- In Sources, where you attach pre-processing Pipelines.
- In Destinations, where you attach post-processing Pipelines.
- In Routes, in the Routing table's **Pipeline/Output** column.

This expanded view shows how a Pack can replace a Pipeline in a Route:



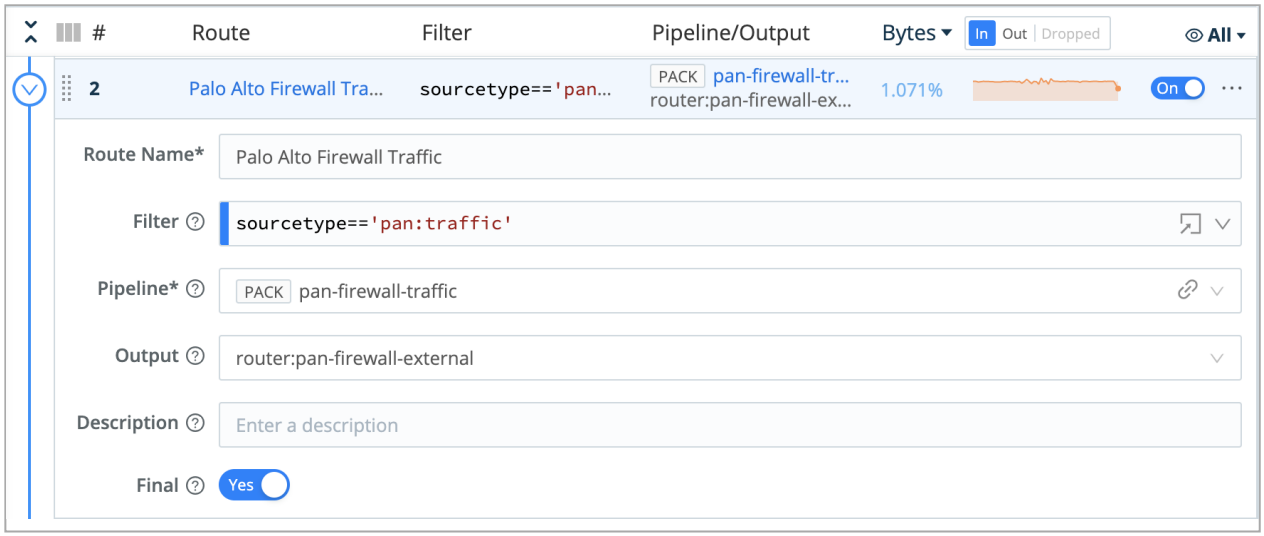
A Pack snaps into Cribl Edge like an enhanced Pipeline

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

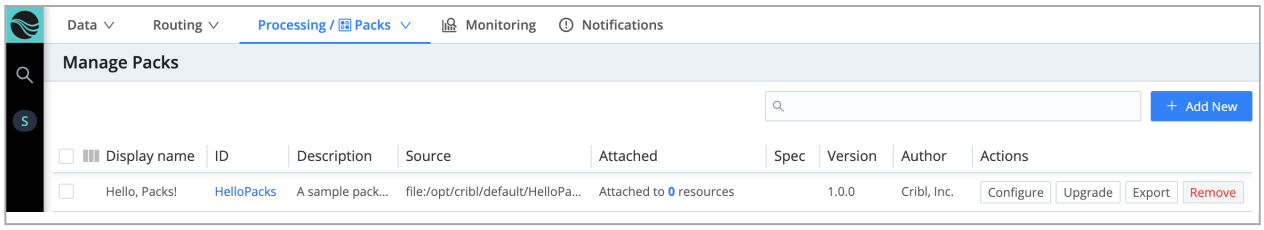
Cribl Edge's **Monitoring** page includes a **Packs** link where you can monitor Packs' throughput.

Accessing Packs

You access Packs differently, depending on your deployment type.

Single-Instance

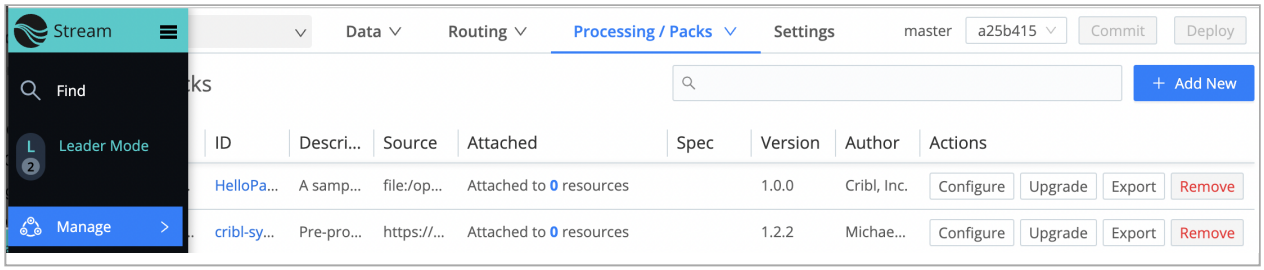
In a single-instance deployment, Packs are global. From Cribl Edge's top-level navigation, just select **Processing** > **Packs**.



Packs, single-instance navigation

Distributed/Default Fleet

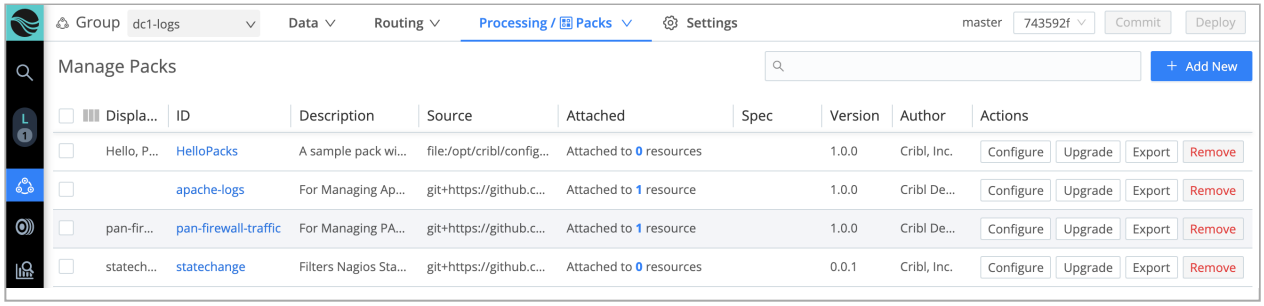
In a distributed deployment with the default single Fleet (Leader mode), select **Manage**, click into the `default` group, and then select **Processing** > **Packs**.



Packs, Leader mode

Distributed/Multiple Worker Groups/Fleets

In a distributed deployment with multiple Worker Groups/Fleets (Leader mode), Packs are associated with (and installed within) Worker Groups. Navigate to the parent Fleet, then select **Processing** > **Packs** from that Group's top nav.



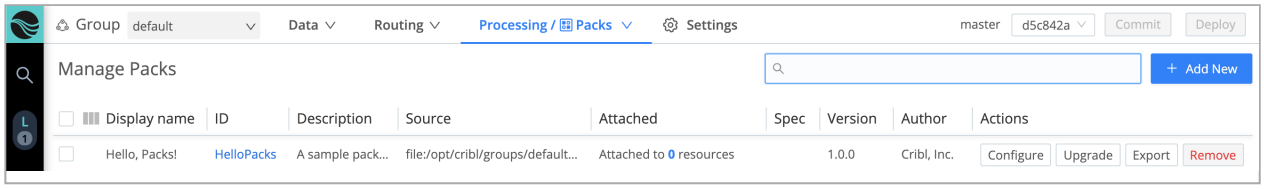
Fleet > Manage Packs page

As the top nav adds more controls on narrower browsers, **Packs** and other right-side links can move onto the ••• overflow menu, as shown above.

> By design, you can readily share Packs **across** Worker Groups/Fleets by exporting/importing them (both covered below).

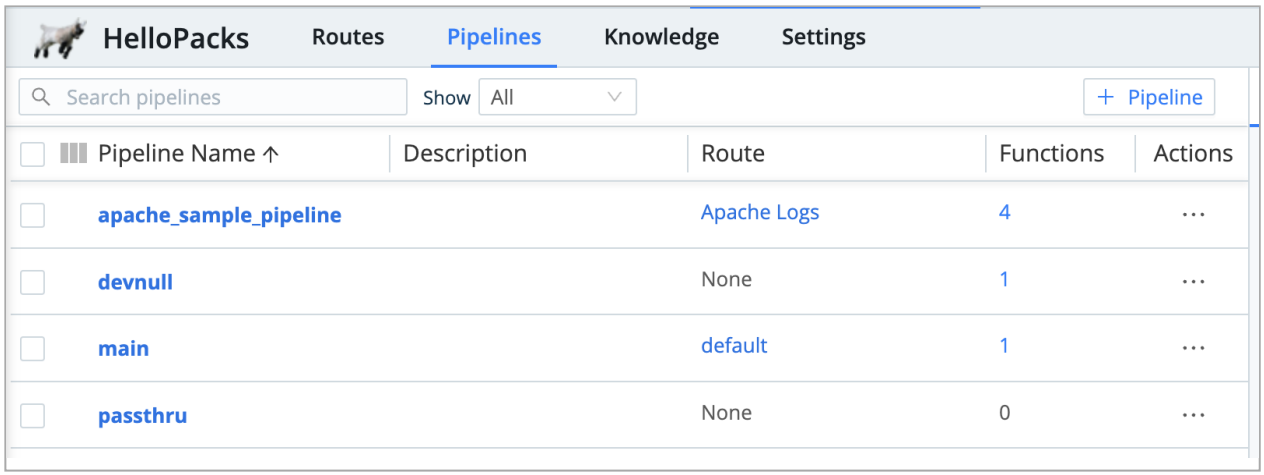
Getting Started with Packs

To unpack Packs, use the above instructions (per deployment type) to navigate to the **HelloPacks** example Pack shipped with Cribl Edge. On the **Manage Packs** page, click this Pack's row to see its configuration.



Manage Packs page with example Pack

Click **Pipelines** on the Pack's submenu, 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 – click it to unpack that, too.

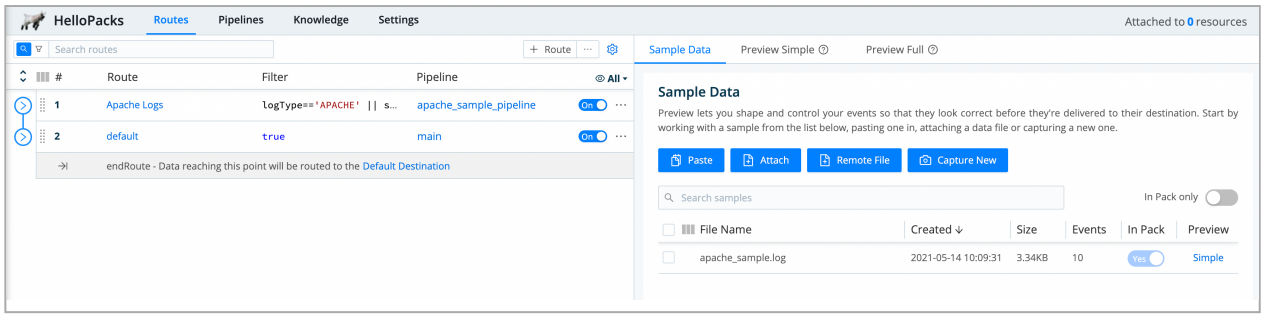


Pipelines within example Pack

Click **Routes** on the Pack's submenu, and you'll see that this Pack also provides both a `default` and an Apache-specific Route.

Configuring a Pack

Once loaded, each Pack displays a submenu with familiar links: **Routes**, **Pipelines**, **Knowledge**, and **Settings** on the left pane, along with **Sample Data**, and **Preview Simple** on the right. The Pack's submenu is a subset of Cribl Edge's top nav.



Configuring a Pack

The left pane's submenu links give you access to configuration objects specific to this Pack.

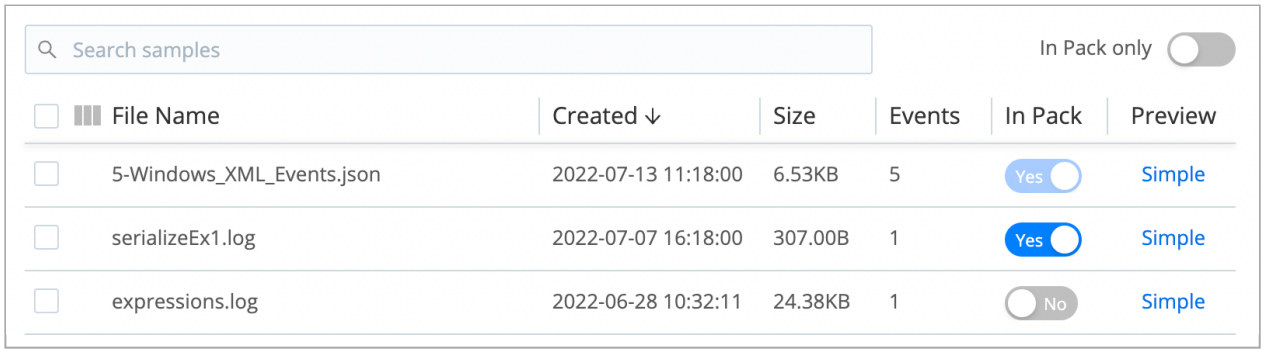
Sample Data

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, toggle the **In Pack only** slider to the right.

If you add sample data files via this 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.



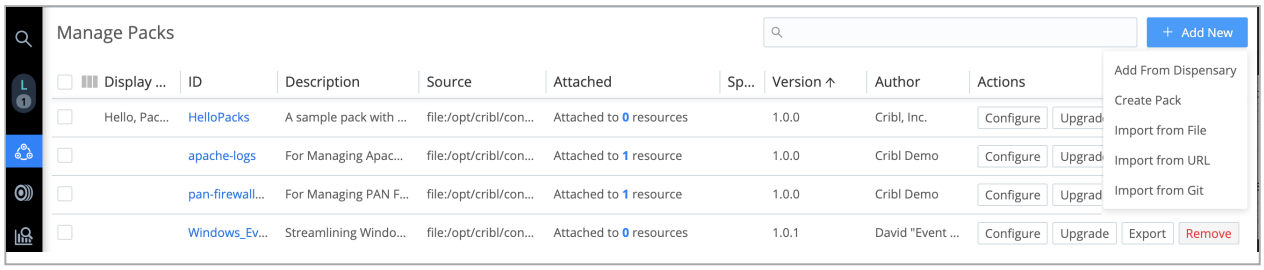
Sample file in Pack

Basically, you can manipulate all the options here as you'd work with their big sister or brother in Cribl Edge's global navigation.

Importing or Upgrading a Pack

To import a new Pack, or an updated version of an existing Pack, from your filesystem:

1. Navigate to the **Manage Packs** page.
2. Click **Add Pack** at the upper right.
3. Select your desired **Add**/**Import** source: Dispensary, File, URL, or Git repo.
4. Follow the above links to details on each of these options.



Importing a Pack

Custom Functions

Packs can include Pipelines containing custom functions, which can (in turn) run arbitrary JavaScript. Before you install a Pack, make sure it comes from a provider you trust, such as the Cribl Packs Dispensary or your own organization.

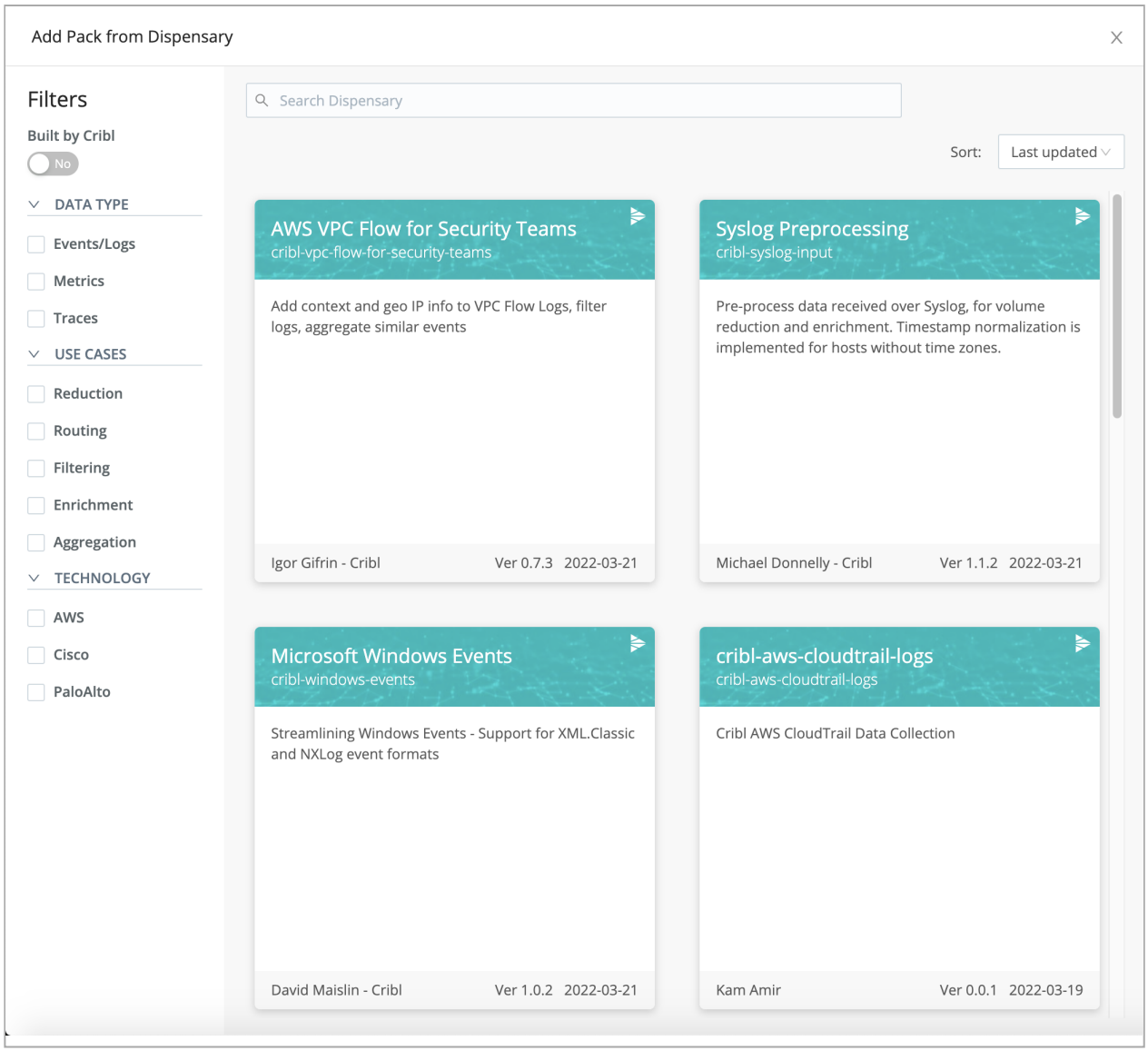
As an additional protection layer, all Pack import modals provide an **Allow custom functions** slider. In the slider's default `No` position, if Cribl Edge detects custom functions in the specified Pack, it will block the import with an error message. If you trust the Pack's provider, toggle the slider to `Yes`, and the import will proceed normally.

The Cribl Packs Dispensary™

You might be wondering, "Where can I find a reliable source of Packs that add useful features to Cribl Edge, vetted for safety?"

Well, Cribl is proud to point you to the Cribl Packs Dispensary™. Here, Cribl's own solutions engineers have seeded several strains of high-productivity Cribl Edge configurations. Because the Packs Dispensary™ is a place to share good stuff, we expect many new hybrids to sprout from the community. Cribl will test and curate submissions to ensure the quality of the repo's contents.

You can install Dispensary Packs directly through Cribl Edge's UI, as outlined in Add from Packs Dispensary below.



Cribl Packs Dispensary™ (as displayed in Cribl Edge's **Add** drawer)

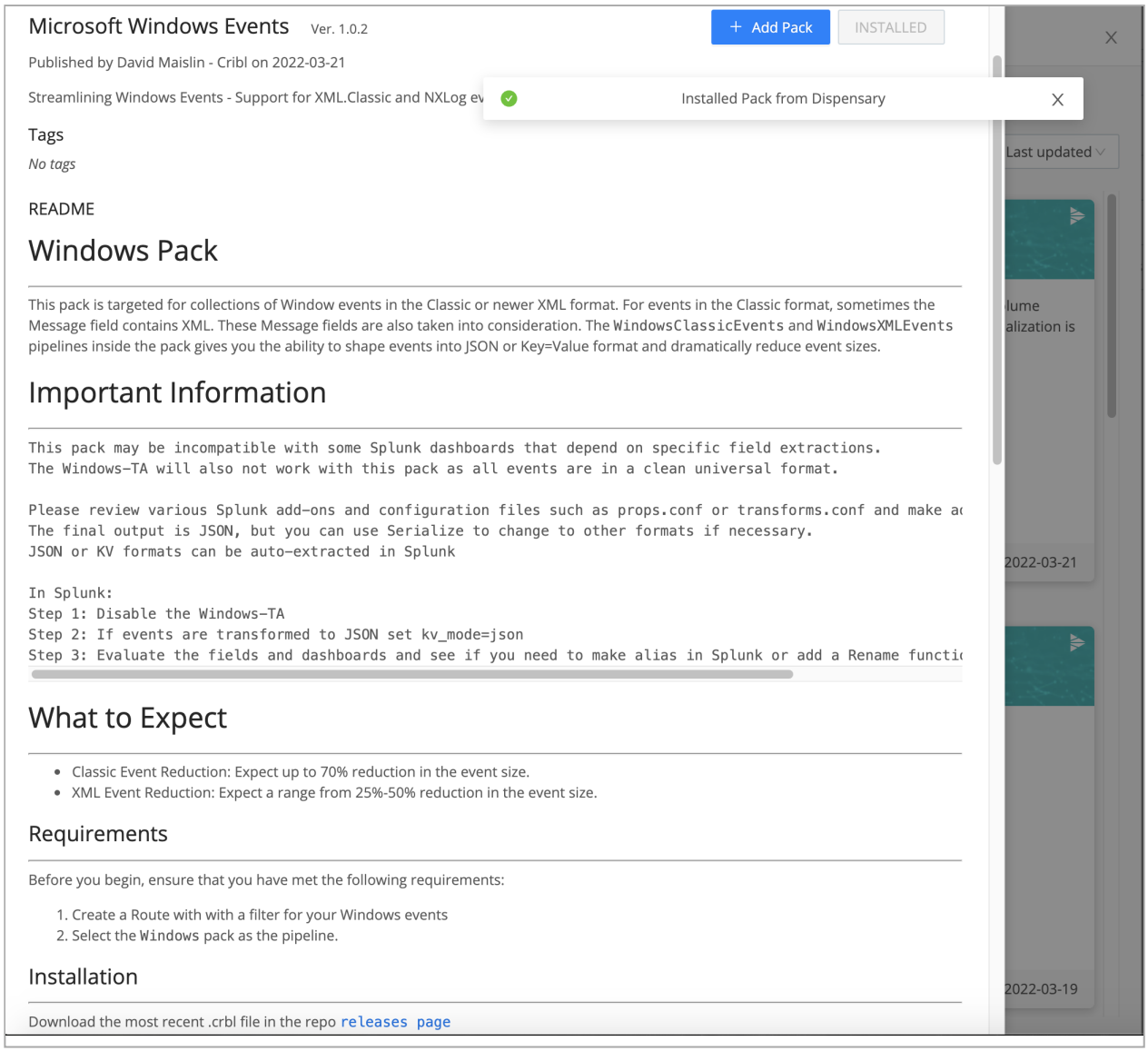
Interested in publishing your own Packs on the Cribl Packs Dispensary™? See Publishing a Pack.

Add from Packs Dispensary

To add a Pack from the Cribl Packs Dispensary™ sharing site:

1. From the **Manage Packs** page's **New Pack** submenu, select **Add from Dispensary**.
2. The Packs Dispensary will open in a drawer, as shown in the screenshot above.
3. Using the drawer's 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.)
4. Click any Pack's tile to display its details page with its README. This will typically outline the Pack's purpose, compatibility, requirements, and installation.

5. To proceed, click **Add Pack** on this page.
6. That's it! You'll see a banner confirming that the Pack is now installed.



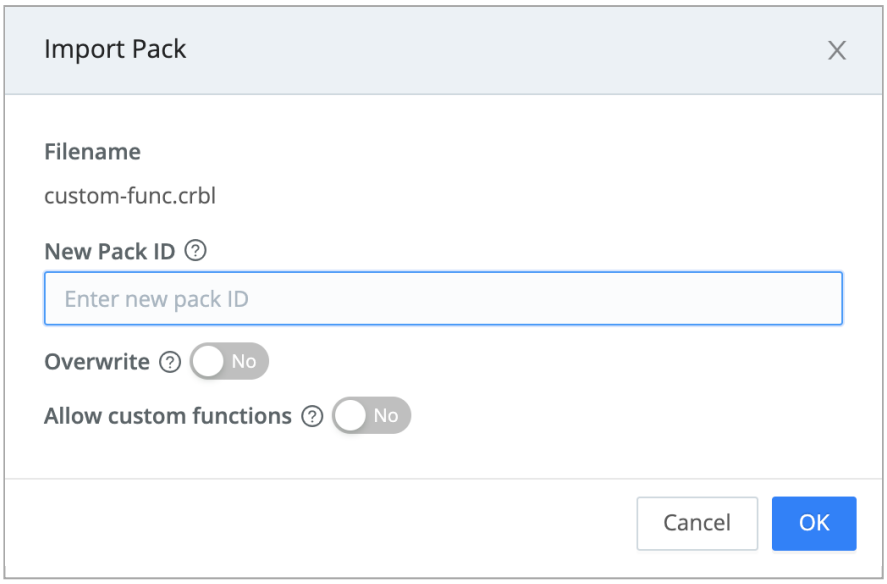
Pack details page: Composite with **Add Pack** button, confirmation banner, Dispensary drawer in background

Import from File

To import a Pack (`.crbl` file) from your local filesystem:

1. From the **New Pack** submenu, select **Import from File**.
2. From the resulting File Open dialog, select the file to import.
3. Optionally, give the pack an explicit, unique **New Pack ID**. (For details about this option, see Upgrading an Existing Pack below.)
4. Where appropriate (see just above), enable **Allow custom functions**.

5. Click **OK** to confirm the import.

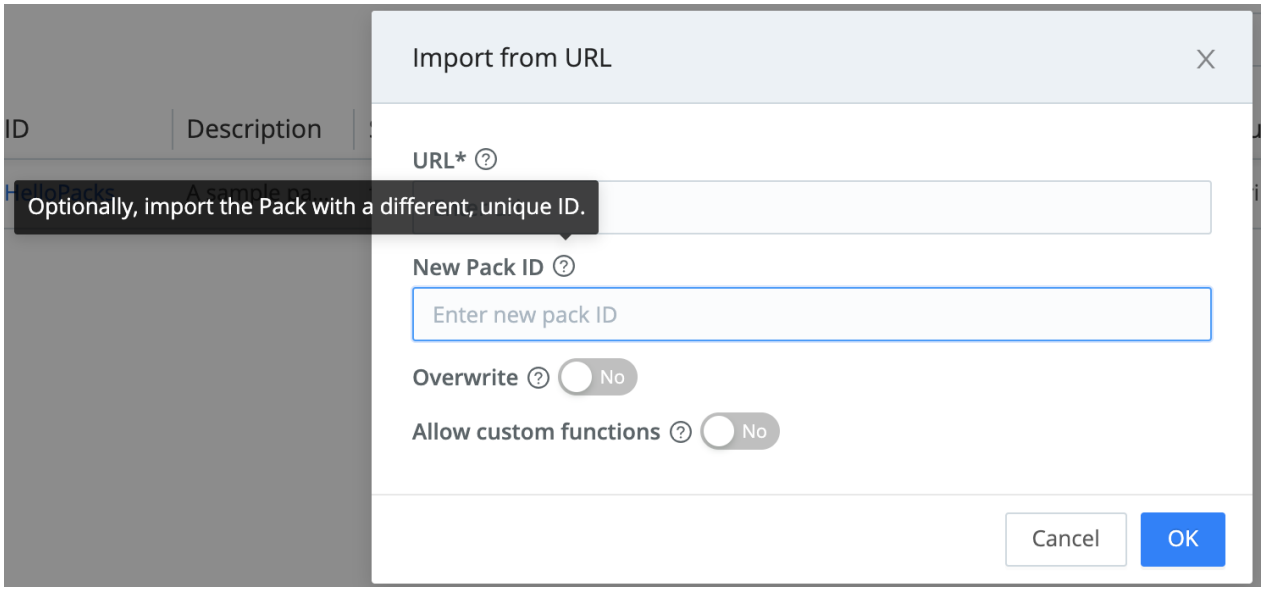


Importing from a file

Import from URL

To import a Pack from a known, public or internal, URL:

1. From the **New Pack** submenu, select **Import from URL**.
2. 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.)
3. Optionally, give the pack an explicit, unique **New Pack ID**. (See Upgrading an Existing Pack.)
4. Where appropriate, enable **Allow custom functions**. (See Custom Functions.)
5. Click **OK** to confirm the import.

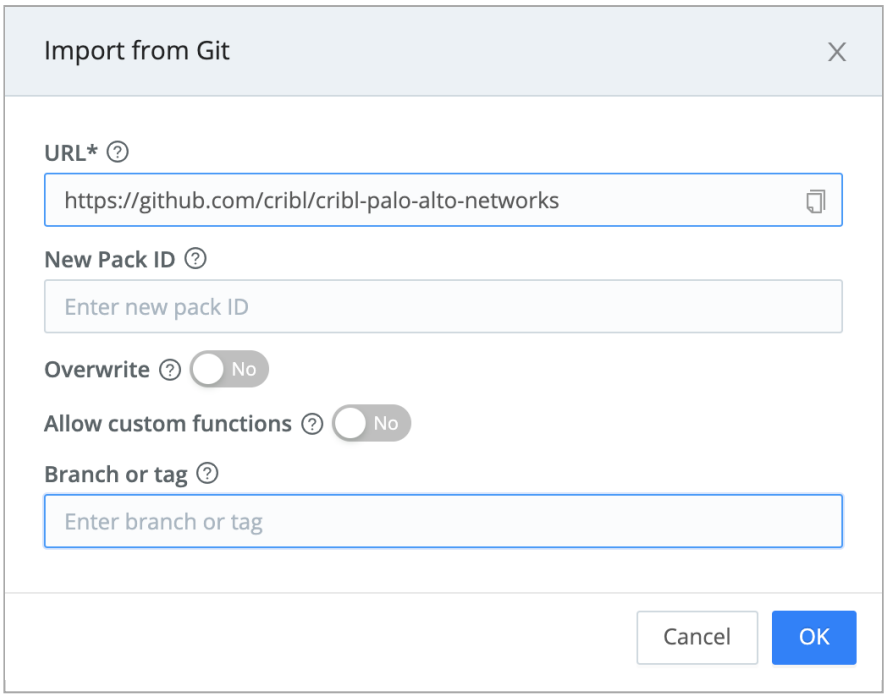


> To import a Pack from a public URL, Cribl Edge's Leader Node (or single instance) requires Internet access. A distributed deployment's Leader can then deploy the Pack to Workers even if the Workers lack Internet access.

Import from Git Repos

To import a Pack from a known public or private Git repo:

1. From the **New Pack** submenu, select **Import from Git**.
2. 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.
3. Optionally, give the pack an explicit, unique **New Pack ID**. (See Upgrading an Existing Pack.)
4. 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`, etc.).
5. Where appropriate (see Custom Functions), enable **Allow custom functions**.
6. Click **OK** to confirm the import.



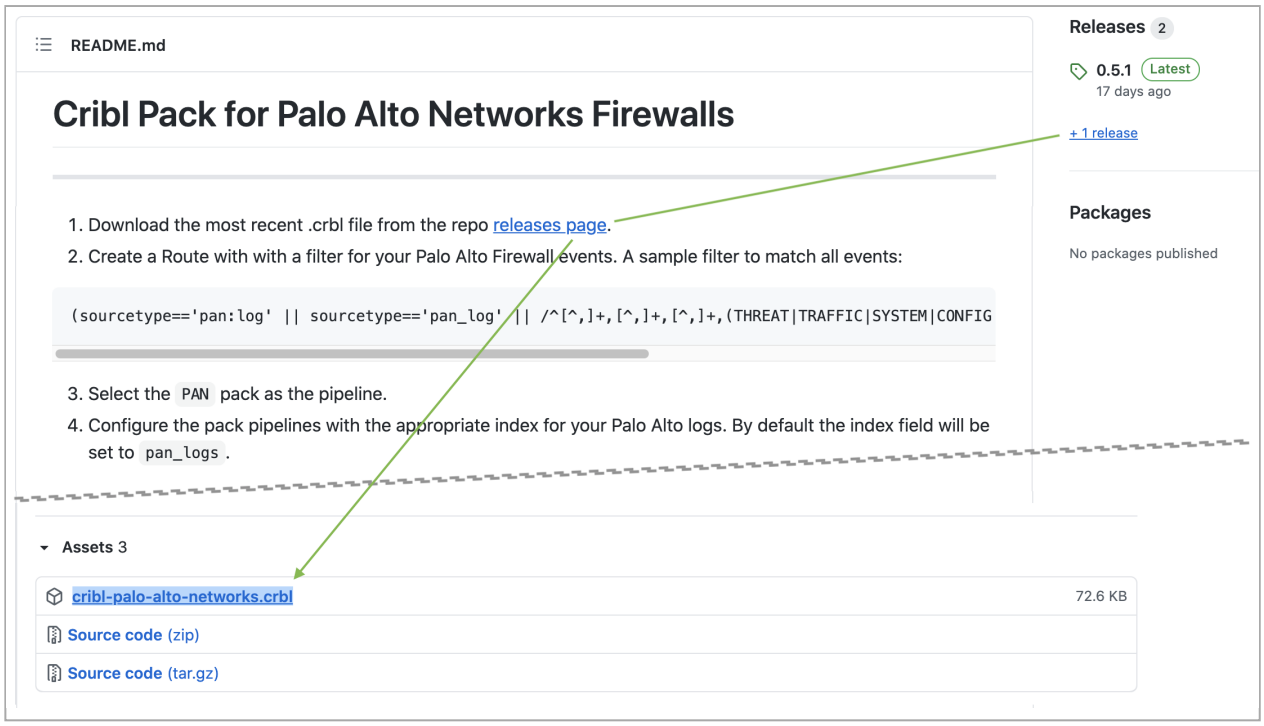
Importing from a Git repo

> To import a Pack from a public repo, Cribl Edge's Leader Node (or single instance) requires Internet access. A distributed deployment's Leader can then deploy the Pack to Workers even if the Workers lack Internet access.

Dispensary GitHub Repo

One authoritative public repo is the Cribl Pack Dispensary on GitHub. (This is the precursor to the Cribl-hosted Cribl Packs Dispensary™ site.)

You can install Dispensary Packs directly through Cribl Edge's UI, as outlined in Import from Git Repos above. However, if you prefer, you can click through to any Dispensary repo's release page, download the corresponding `.crbl` file, and then upload the file into Cribl Edge.



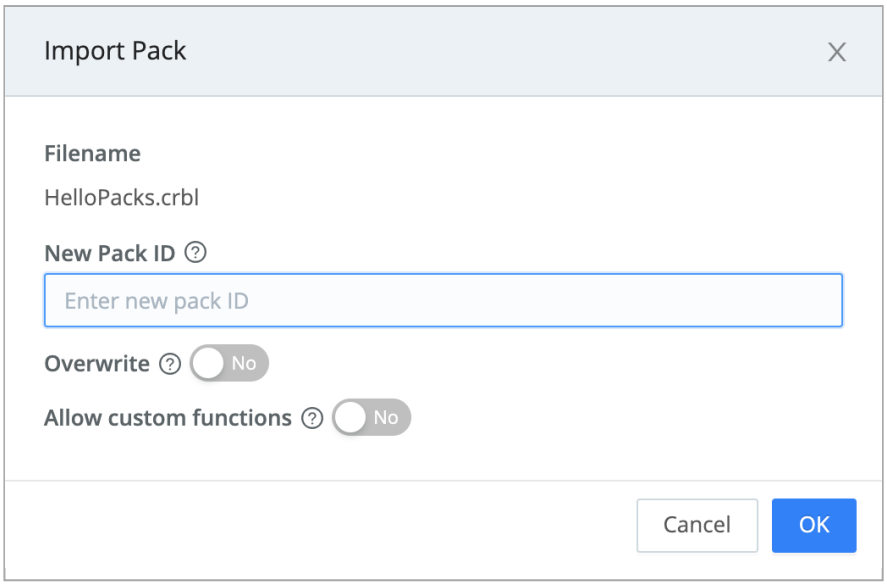
Downloading a `.crbl` file from the Cribl Pack Dispensary's Web UI

If you've posted completed Packs to our GitHub repo, we encourage you to now submit them to our new Cribl Packs Dispensary™ site. See Publishing a Pack.

Upgrading an Existing Pack

Each Pack that is installed within a given Fleet (or single-instance deployment) must have a unique ID. The ID is based on the Pack's internal configuration – not its container's 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.



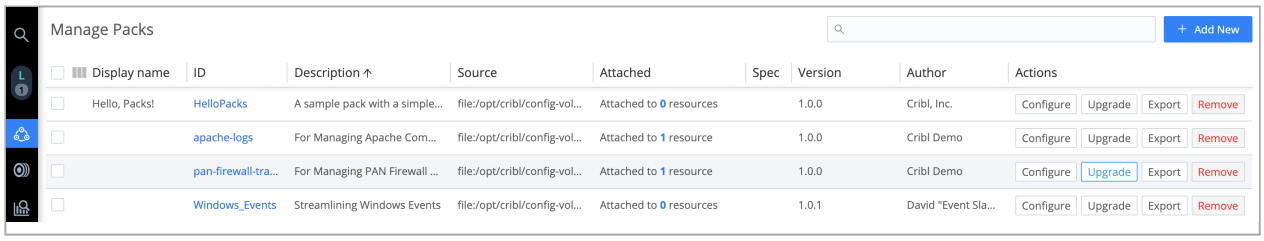
Renaming a Pack on import

You'll also have the option to **Overwrite** the installed Pack, reusing the same ID.

If you toggle this option to `Yes`, the imported Pack will completely overwrite your existing Pack's 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 click the **Upgrade** button on its row.



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:

- Import the updated Pack under a new name that includes the version number (e.g., `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 Knowledge objects (e.g., Lookup files), be sure to copy the modified files locally for safe keeping before upgrading the Pack. After installing the upgrade, copy those files over the Pack upgrade's default versions.
- Test, test, and test!
- Commit and Deploy.

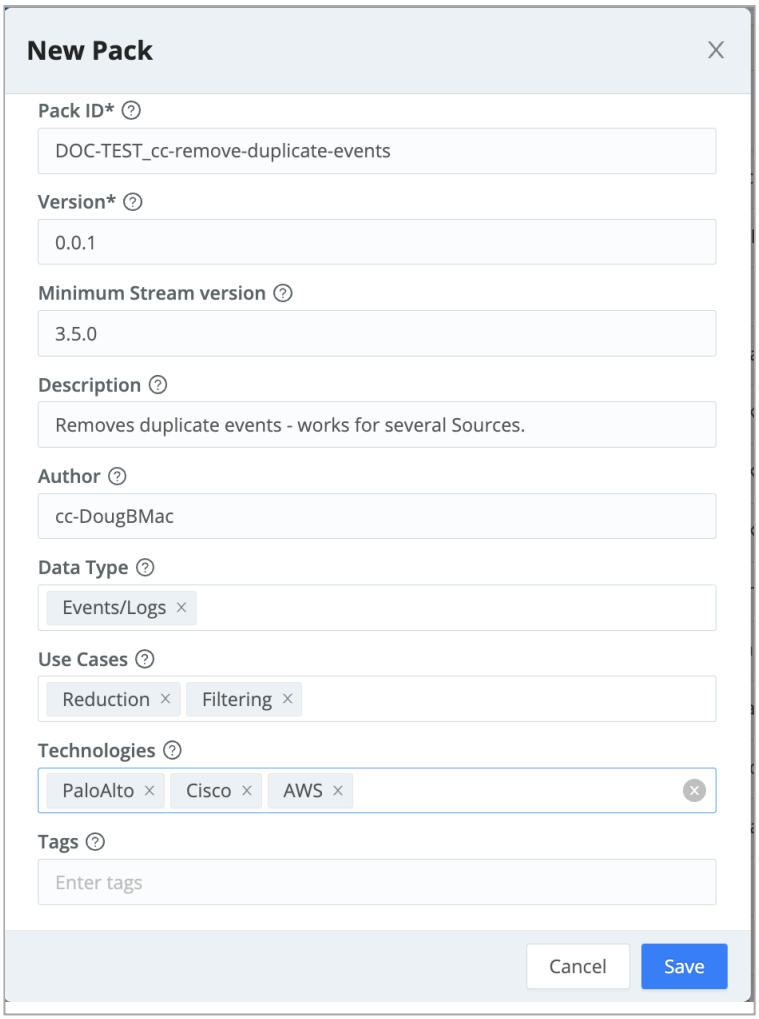
Creating a Pack

You can create a new Pack from scratch, to consolidate and export multiple Cribl Edge configuration objects:

1. Navigate to the **Manage Packs** page.
2. Click **Add Pack**.
3. From the submenu, select **Create Pack**.
4. In the resulting **New Pack** modal, fill in a unique **Pack ID** and other details.

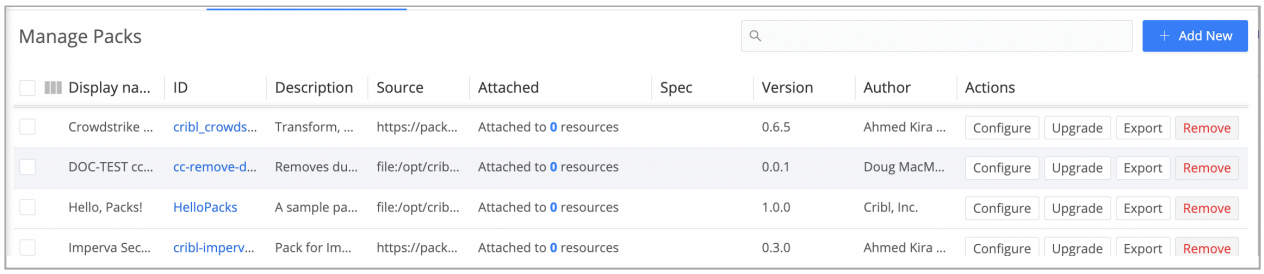
   - Each Pack within a Cribl Edge instance must have a separate **Pack ID**, but you can assign arbitrary **Display name**s.
   - **Version** is a required field identifying the Pack's own versioning.
   - **Minimum Stream version** is an optional field specifying the lowest compatible version of Cribl 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.

5. Click **Save**.



Creating a Pack

6. On the **Manage Packs** page, click the new Pack's row to open the Pack.



Manage Packs page

7. Use the standard Cribl Edge controls (see above) to configure and save the infrastructure you want to pack up. As you save changes in the UI, they're saved to the Pack.

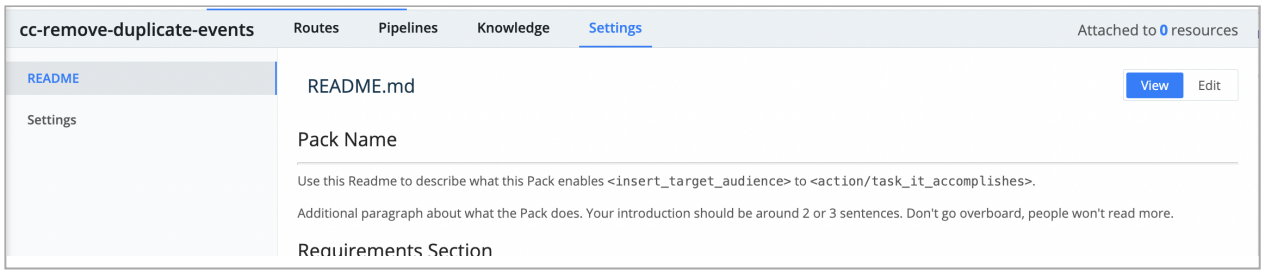
> If you'd like to share your Pack with the community of Cribl users, you can publish it on the Cribl Packs Dispensary™.

> 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 Crlbl.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!

Modifying Pack Settings

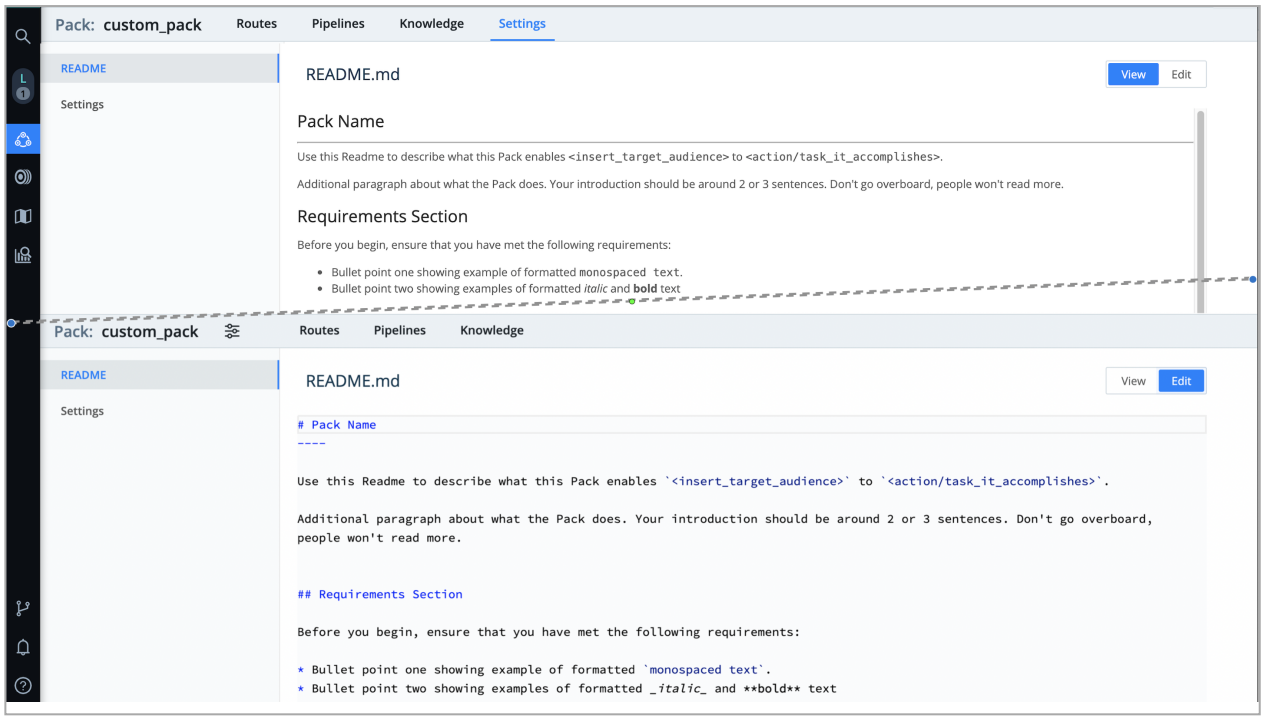
You can update a Pack's metadata (Version, Description, Author, etc.) and display settings. If you're developing a new Pack to share, you'll want to use this interface to populate the Pack's README and display logo.

1. From the Pack's submenu, select **Settings**. The left **README** tab will gain focus.



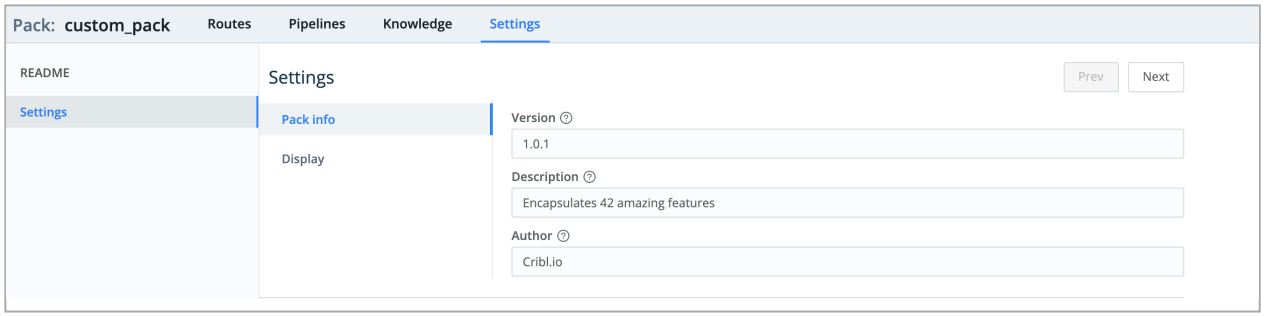
Pack Settings

2. To populate the Pack's README file, toggle **View** to **Edit**, replace the placeholder markdown content, and **Save**.



Editing Pack's README

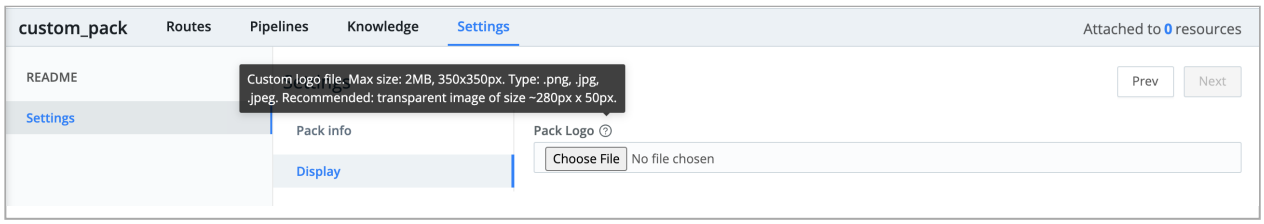
3. To update other metadata, click the left **Settings** tab.



Editing Pack's metadata

4. To add a Pack logo, click the Pack's **Settings** > **Display** left tab.

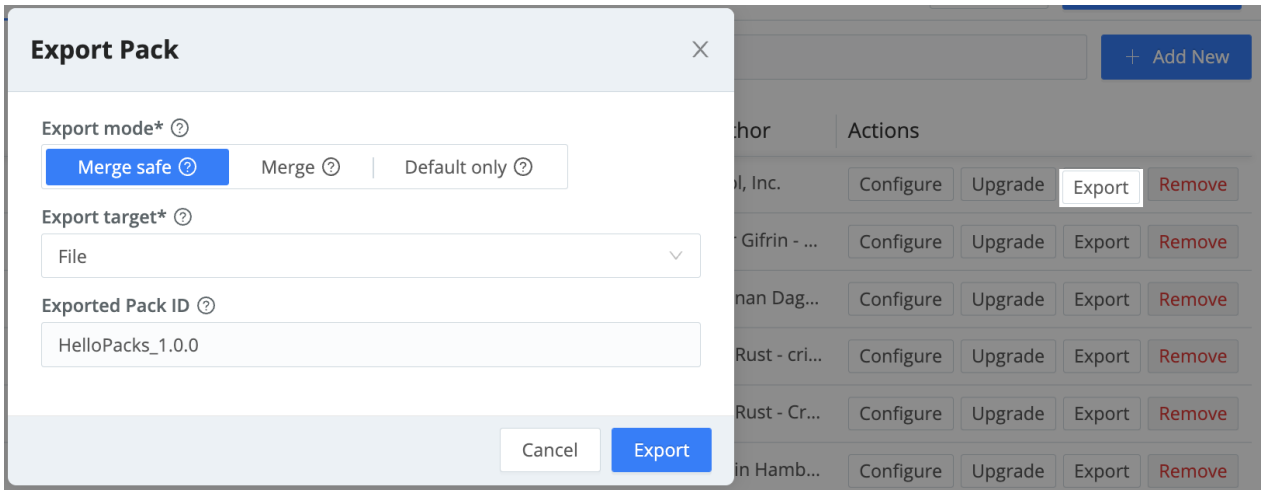
   Cribl recommends adding a logo to each custom Pack, to visually distinguish the Pack's UI from the surrounding Cribl Edge UI (as well as from other Packs). You can upload a `.png` or `.jpg`/`.jpeg` file, up to a maximum size of 2MB and 350x350px. Cribl recommends a transparent image, sized approximately 280x50px.



Editing Pack's display (logo) settings

Exporting a Pack

To export a newly created or modified Pack, click its **Export** button on the Packs page.



Exporting a Pack

The resulting Export Pack modal provides the following options.

Export Mode

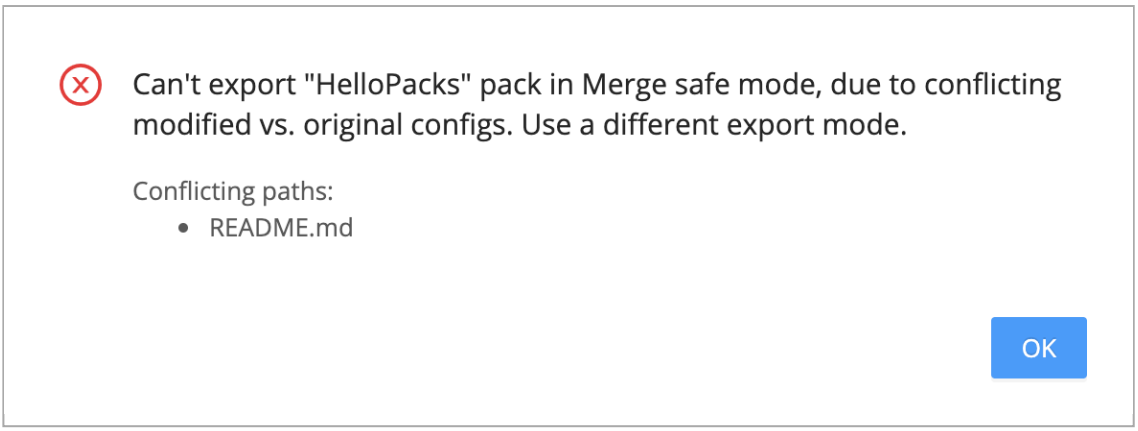
Select one of these three buttons:

- **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.

  > **Merge** is the only export mode available when you've selected `Groups` as your Export target.

- **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 conflicting, modified contents with the Pack's original contents:



Merge safe error

If you encounter an error like the example shown above, use the **Merge** or **Default only** export mode instead.

Export Target

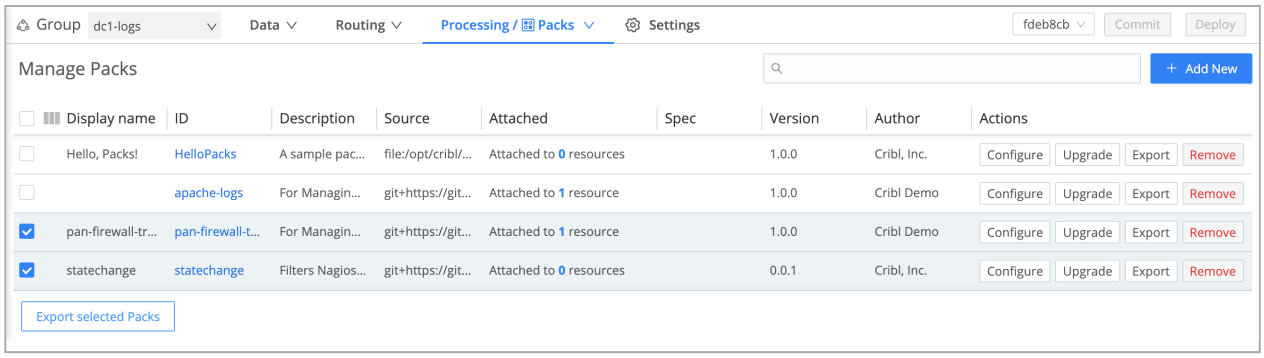
The options here are:

- **File** (the default): You'll be prompted to confirm a file name and destination after you click **OK**. (In Cribl Edge 3.5 and higher, the default file name automatically includes the Pack's version number.)
- **Groups**: Selecting this displays a **Groups** control, prompting you to select one or multiple existing Worker Groups/Fleets to export the Pack to. (The current Fleet is automatically omitted from the options.)

Exporting Multiple Packs

You can export multiple Packs in one operation, by selecting their check boxes and then clicking **Export multiple Packs**. This option comes with a few constraints:

- You can export multiple Packs only to Groups (not to Files).
- Therefore, this option is available only in distributed deployments.
- The only export mode available is **Merge**.
- The **Exported Pack ID** field is disabled, and hidden.



Exporting multiple packs at once

A status modal will list any Packs that failed to export.

Exported Pack ID

Defaults to the Pack's current ID, with the version number appended. This is an opportunity to change the Pack's ID if you're exporting it as you develop a new version.

Managing Packs via API

You can perform Pack operations by running Cribl Edge API calls on the command line. This is required if you plan to automate Pack operations, e.g., 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/packs/goat-herd/export?mode=merge' > goat-
herd.crbl
```

This request returns an octet-stream attachment which is downloaded as a `.crbl` file. And voilà, 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 -H "Authorization: Bearer $token"
'https://stream:9000/api/v1/m/group420/packs?filename=goat-herd.crbl' --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, the `goat-herd` Pack is renamed 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 -H "Authorization: Bearer $token" -H "Content-Type: application/json"
'https://stream:9000/api/v1/m/group420/packs' -d '{"source":"goat-
herd.987654321.crbl", "id":"billys_pack"}'
```

Copy via API

To bulk-copy Packs between Worker Groups/Fleets, adapt and run this **Clone pack** API call, referencing a source Fleet, destination Fleet(s)/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"
'https://stream:9000/api/v1/packs/__clone__' -d '{"srcGroup":"default","dstGroups":
["dc1-logs","dc2-logs"],"packs":["PAN","cribl_cisco_asa_cleanup"]}'
```

Last updated by: Dritan Bitincka

11.5.1. 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 three-step process:

1. Prepare and Produce.

   In this initial phase, feel free to share with other Community members, who can help refine the Pack. For this development phase, consider working collaboratively within Cribl's Community Slack `#packs` channel, or on a publicly available Git repo that you own and maintain - for an example of how to use Git to create and manage Packs, see Cribl's Dispensary GitHub project.

2. Publish the Pack to the Cribl Packs Dispensary™.

   The submission process, outlined below, validates that all required fields are included: Pack name, version, author, and license, if the version is newer than the last one published.

3. Celebrate!

> If you've already posted completed Packs to Cribl's GitHub repo, we encourage you to submit them to the Packs Dispensary™. See Publishing a Pack.

Community Pack Guidelines

A Cribl Community Pack must be useful, reusable, and subject to the Cribl Pack Developer Agreement (PDA).

Making Your Pack Useful

Above all, what makes a Cribl Community Pack useful is the value that it provides for your fellow Community members. A Pack will be most useful if it includes Pipelines, along with supporting sample files and Knowledge objects (especially Lookups) as needed.

Making Your Pack Reusable

Reusability means that the Pack brings value to multiple Cribl Edge users. To make this possible, you should be sure your README provides:

1. Instructions for using the Pack, including details on how to configure any relevant Sources and Destinations.

2. Details about the impact on downstream systems, so that users can prepare for changes that the Pack will make to data flowing through.

Acknowledging the Pack Developer Agreement

The Pack Developer Agreement (PDA) appears when you first access the Cribl Packs Dispensary™. As a Pack author, you must electronically acknowledge that you have read the Agreement, and that you intend to adhere to its requirements.

Before you Begin

Before you start creating your Pack, check the Cribl Packs Dispensary™ to see if something similar has already been published. If your idea for a Pack is new:

- Post to Cribl Community Slack's the `#packs` channel, asking whether any of your fellow Community members are working on a Pack that's similar to yours.

- If someone is already working along the same lines, then you have a good opportunity to collaborate on the Pack you want to create.

- Read the docs that explain how to create your Pack from within the Cribl Edge UI.

Creating the Pack

Here's how you should formulate the information you'll need to create the Pack.

Pack names **should not**:

- Start with `Cribl` or `Cribl-`, which are reserved for Cribl-created packs.
- Use the word `Pack`, at all.

Pack names **should**:

- Start with `cc` (this indicates that the Pack was contributed by a Cribl Community member).
- Use all lowercase.
- Use dashes to separate words, e.g., `cc-tanium-events`.

Pack Version numbers should:

- Use `0.1.0` for the initial development version of the Pack.
- Designate (number) subsequent versions as described in Publishing a Pack.

Pack Descriptions should:

- Be brief - no more than 1 or 2 sentences, e.g., `This Pack for Syslog inputs will reduce volume, and address timestamp normalization for Syslog senders that omit timezones.`

Pack Author names should be in the following format:

- Your Community name, a dash (`-`) and `cc`, e.g., `art chavez - cc`.

Pack License Notes should:

- Appear at the end of the Pack `README` and appropriately linked, e.g., `This Pack uses the following license: [Apache 2.0](https://github.com/criblio/appscope/blob/master/LICENSE).`

Creating Documentation for the Pack

To properly scope the documentation you write for your Pack, follow these principles:

- The optimal user experience is to have a single Pack that supports all datasets for the relevant device or sender.
- The Pack documentation should list all of the datasets that the Pack supports.
- If there are known datasets that are relevant but not yet supported, the documentation should list those, too.

For example, consider a device sending data to Cribl Edge, such as a Palo Alto Networks firewall. This device supports multiple datasets, in that Palo Alto Firewall data is really one co-mingled set of data that includes individual datasets like `PAN-Traffic`, `PAN-System`, `PAN-Accept`, and so on.

Think about what datasets would be involved for other devices, and how you would document them. For example, consider an F5 load balancer, various types of routers, or various types of servers. This exercise will help you anticipate what your users will need to see documented for your own Pack.

What a Pack Must Contain

Every Pack must contain some combination of Pipelines, samples, Routes, Knowledge objects (including Lookups), configuration descriptions, support contact information, and release notes. The exact ingredients will vary by Pack.

Pipelines

1. Except for rare cases, each Pack should have at least one Pipeline. Packs without Pipelines are of limited use.

2. Cribl recommends that you provide one Pipeline (and one Route) for each dataset that your Pack supports.

3. Each Pipeline should have an internal **Comment** describing the overall functionality **and benefits** that the Pipeline provides. The procedure for adding a **Comment** is the same as for adding any other Function. **Comment** is under the **Add Function** > **Standard** drop-down.

4. Within each Pipeline, follow these best practices for Functions:

    - Use grouping to bundle any Functions that a user should enable or disable together.
    - Add a **Description** to each Function, to make it clear what is happening at each step, along with the Function's purpose and mechanics.

5. Overall, design for supportability and efficiency.

Samples

1. Include at least one data sample for each Pipeline. Data samples can be reused across Pipelines, but it is preferable to include multiple data samples, each specific to a Route/filter available in the Pipeline.

2. Remember to remove all sensitive data (e.g., internal host names or IP addresses) from samples.

Routes

1. A Pack includes Routes with appropriate filters.

2. Each Pipeline should have a corresponding Route on the Pack’s Routes page. (The only exception to this is when the Pack serves as a delivery mechanism for pre-processing and post-processing Pipelines.)

3. The filter in that Route should be as generic as possible. For example, if the data is coming from `PAN`, don't assume you'll have `sourcetype`. Instead, filter by `_raw.match()`.

4. Each Route must have a meaningful description.

5. The final Route (where **Filter** is set to `true`) should route to the `devnull` Pipeline.

Knowledge Objects

1. Include within the Pack all Knowledge objects that the Pack requires – Lookups, Parsers, Global Variables, Grok Patterns, and Schemas.

2. Remember to remove all sensitive data from Knowledge objects – e.g., internal host names and IP addresses.

3. Lookups are especially important. If it's not practical to include the required Lookup, include instructions for building it.

Configuration Descriptions

Where applicable, the Pack documentation should include clear descriptions for each pre-shipped configuration.

Support Contact Information

The Pack documentation should list your preferred method of contact for support, e.g., your Cribl Community name for Slack DMs, or your email address.

Release Notes

If the Pack is an update from a previous release, the documentation should include Release Notes.

What a Pack Should Contain

A logo and a `README` file are recommended, but optional.

The Logo

You can include a logo associated with the technology addressed by the Pack, e.g., a Windows logo for Windows events, an AWS Cloudwatch logo for AWS Cloudwatch data, and so on.

The README

Including a `README` improves the user experience for your Pack. Create a `README.md` file from the Pack's **Settings** submenu, with detailed answers to the following questions:

1. What does the Pack do, and why was it created? Here, you can state what value the Pack provides.
2. What technologies, data sources, and data destinations does the Pack interact with?
3. What other dependencies does the Pack have? As examples:
    - Does the Pack use an external tool, like Redis?
    - What is the minimum supported version of Cribl Edge? This must be newer than Cribl LogStream v.3.0.4 or Cribl Edge 3.3.0.
    - What deployment restrictions apply? What combination of single instance, distributed, or Cloud deployments of Cribl Edge does the Pack support?
4. What is required to configure a data source or destination?
    - Include specific examples of configurations when possible.
    - Link to specific sources and destinations to configure, e.g., link to AWS Firehose Source for a Pack that requires Firehose to collect data.
5. What else does your user need to know to use the Pack?
    - Include specific instructions.
6. How should your users contact you for support?
    - The `README` should contain the Pack author's contact info for providing feedback/requesting support, e.g., your Community name or email address.

Versioning Packs

Packs start at version `0.1.0`, and continue through as many "pre-release" versions as needed, until the authors feel that the Pack is ready for production use. During this initial **Prepare** phase, you'll share your idea with other Cribl Community members, to collaboratively refine the Pack.

Next, you'll enter the **Publish** phase, where it's appropriate to release version `1.0.0`. Here, you'll:

- Satisfy all the requirements for publishing a Pack, as documented above.
- Export the Pack by creating a `.crbl` file.
- Upload the Pack to the Cribl Packs Dispensary™, as outlined below. The Dispensary will validate the Pack name, version, author, and license, including by checking that the new version number is greater than the last one.

Versioning an Existing Pack

The easiest way to do this is to make the Pack changes in Cribl Edge, then export the new Pack and upload it to the Cribl Packs Dispensary™.

1. Update your Pack.
2. Increment the version number. See semantic versioning for guidelines.
3. In the `README`, update the **Release Notes** section to specify the new version and release date, and to describe what has changed. For a good model of how this is done, see the `README` on the Dispensary's **Microsoft Windows Events** Pack.
4. Export your Pack and save the `.crbl` file locally.
5. Upload your Pack's `.crbl` file to the Cribl Packs Dispensary™, as outlined just below. Remember to version the file.

Publishing a Pack

To submit your Pack:

1. Sign into, or create an account on, the Cribl Packs Dispensary™ site. You can create an account using the same email address as used on your Cribl.Cloud account.

   If you don't have a Cribl.Cloud account, Cribl automatically creates a new one for you when you create your account on the Cribl Packs Dispensary™.

2. Once signed in, you'll see the **Publish Pack** and the **View only my Packs** controls shown below.

Packs Dispensary™: Signed-in view

3. Click **Publish Pack**, and then click **Upload Pack**.

4. Select the `.crbl` Pack file you want to publish, and click **Submit**.

5. The Packs Dispensary™ will quickly verify whether the Pack has valid configurations and whether it meets all of the requirements outlined in this document.

6. If validation is successful, the Dispensary submits the Pack for review, and graduates it to a `Validating` state.

7. Once your Pack is reviewed, you'll receive an email from `packs@cribl.io` informing you whether it was accepted or rejected.

   Rejected Packs display on the Packs Dispensary™ as `Failed`, with a note about the rationale for rejection. After you've fixed your Pack, you can resubmit it.

Who Supports Packs

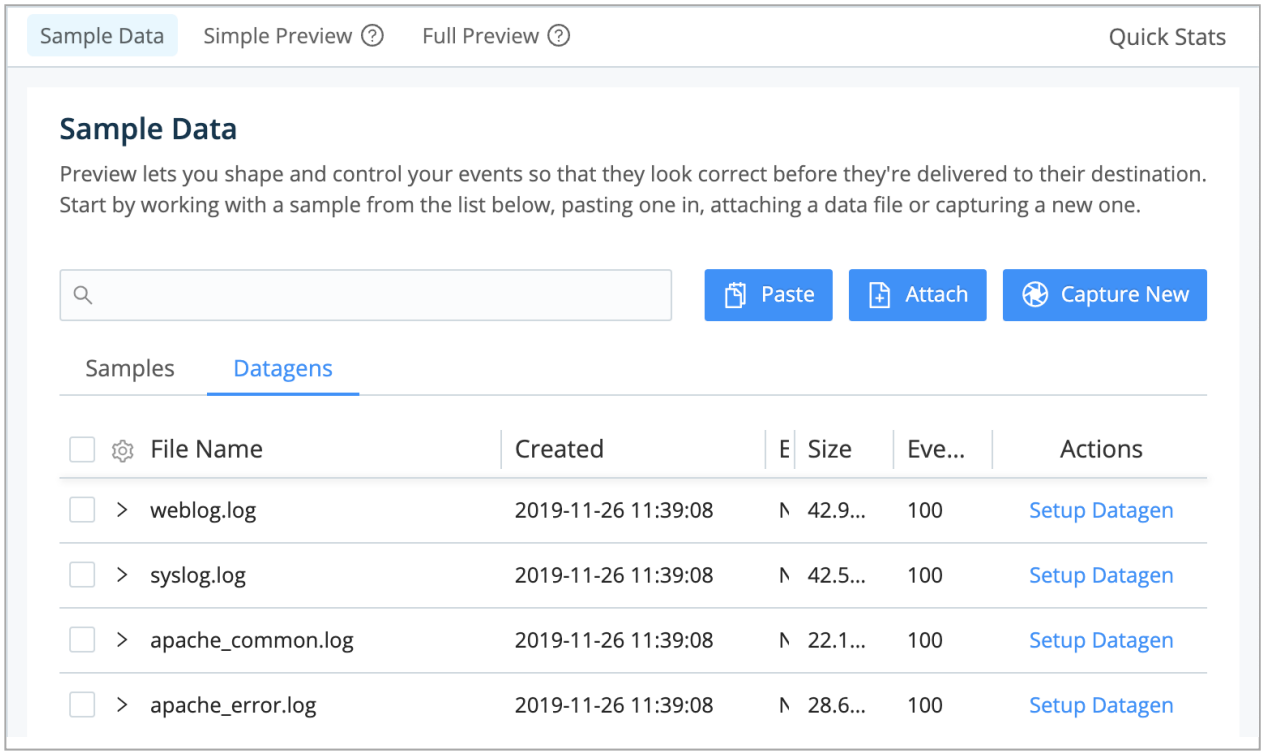
- Community-authored Packs are primarily supported by the Pack author, who also handles feature requests and suggestions.
- Cribl-authored Packs are supported by Cribl.
- For both kinds of Packs, the greater Cribl Slack Community also provides a wealth of knowledge - see the `#packs` channel.

11.6. Using 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

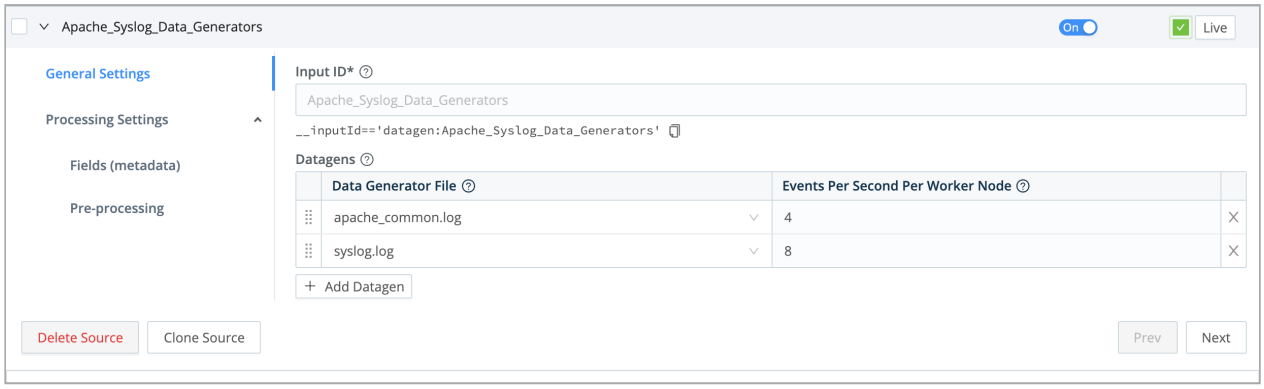
As outlined in the following tutorial: Once you've created a template, you can configure a Datagen Source to use the template to generate real-time data at a given EPS (events per second) rate.

Enabling a Datagen

To see how datagens work, start by enabling a pair of Cribl Edge's out-of-the-box generators:

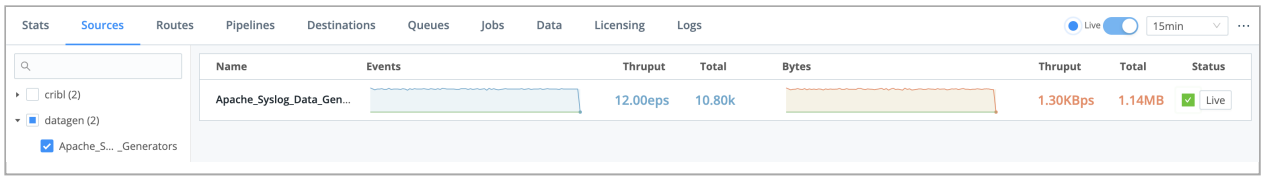
Navigate to **Sources** > **Datagens** and click **New Destination**.

Select a Data Generator File (e.g., `apache_common.log`) and set it at 4 EPS/worker process. Select another Data Generator File (e.g., `syslog.log`) and set it at 8 EPS/worker process. Hit **Save**.



Selecting datagens files and event rates

On the **Monitoring** page, under **Sources**, search for `datagen` and confirm that the Source is generating data.



Creating a Datagen Template from a Sample File

To convert a sample into a template:

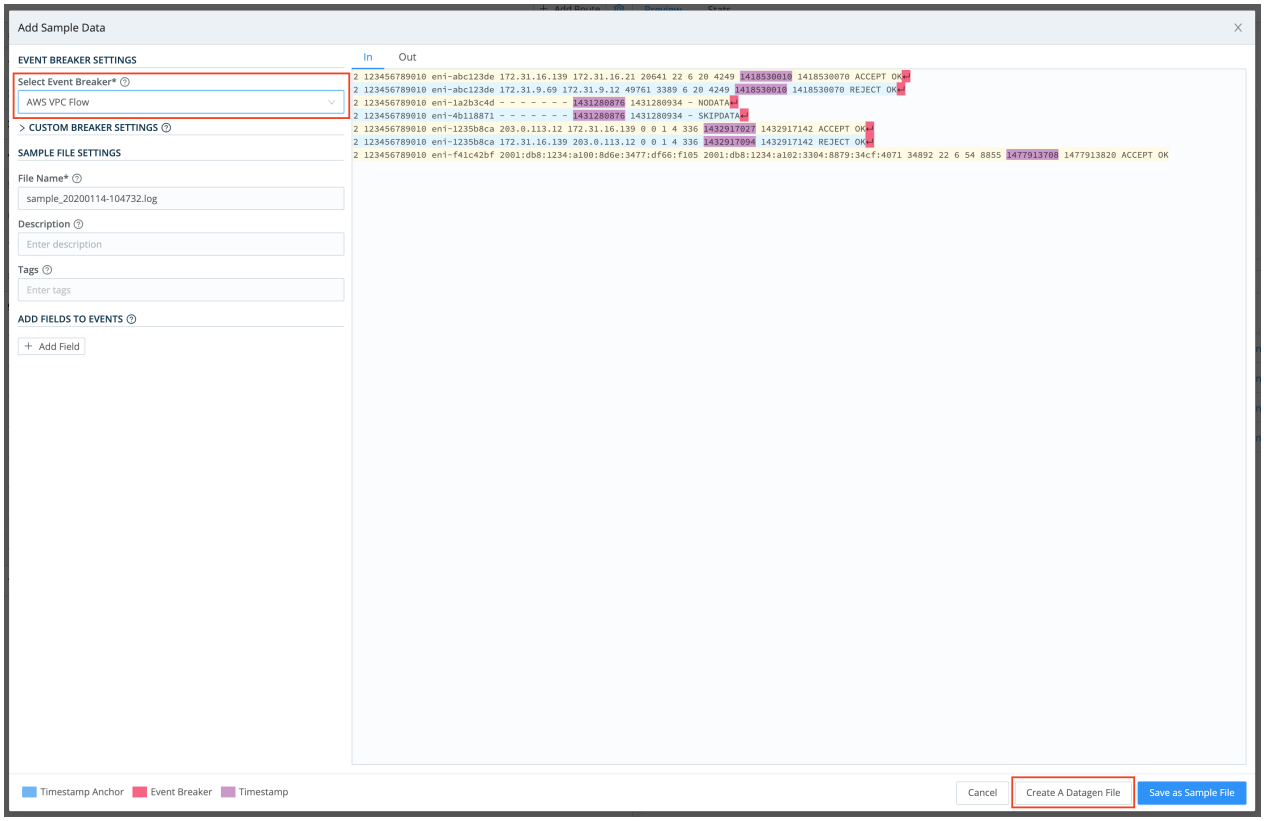
Go to **Preview** > **Paste a Sample**, and add a sample like the AWS VPC Flow logs below:

```
2 123456789010 eni-abc123de 172.31.16.139 172.31.16.21 20641 22 6 20 4249 1418530010
1418530070 ACCEPT OK
2 123456789010 eni-abc123de 172.31.9.69 172.31.9.12 49761 3389 6 20 4249 1418530010
1418530070 REJECT OK
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
1432917142 ACCEPT OK
2 123456789010 eni-1235b8ca 172.31.16.139 203.0.113.12 0 0 1 4 336 1432917094
1432917142 REJECT OK
2 123456789010 eni-f41c42bf 2001:db8:1234:a100:8d6e:3477:df66:f105
2001:db8:1234:a102:3304:8879:34cf:4071 34892 22 6 54 8855 1477913708 1477913820
ACCEPT OK
```

From the **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, click **Create a Datagen File**.

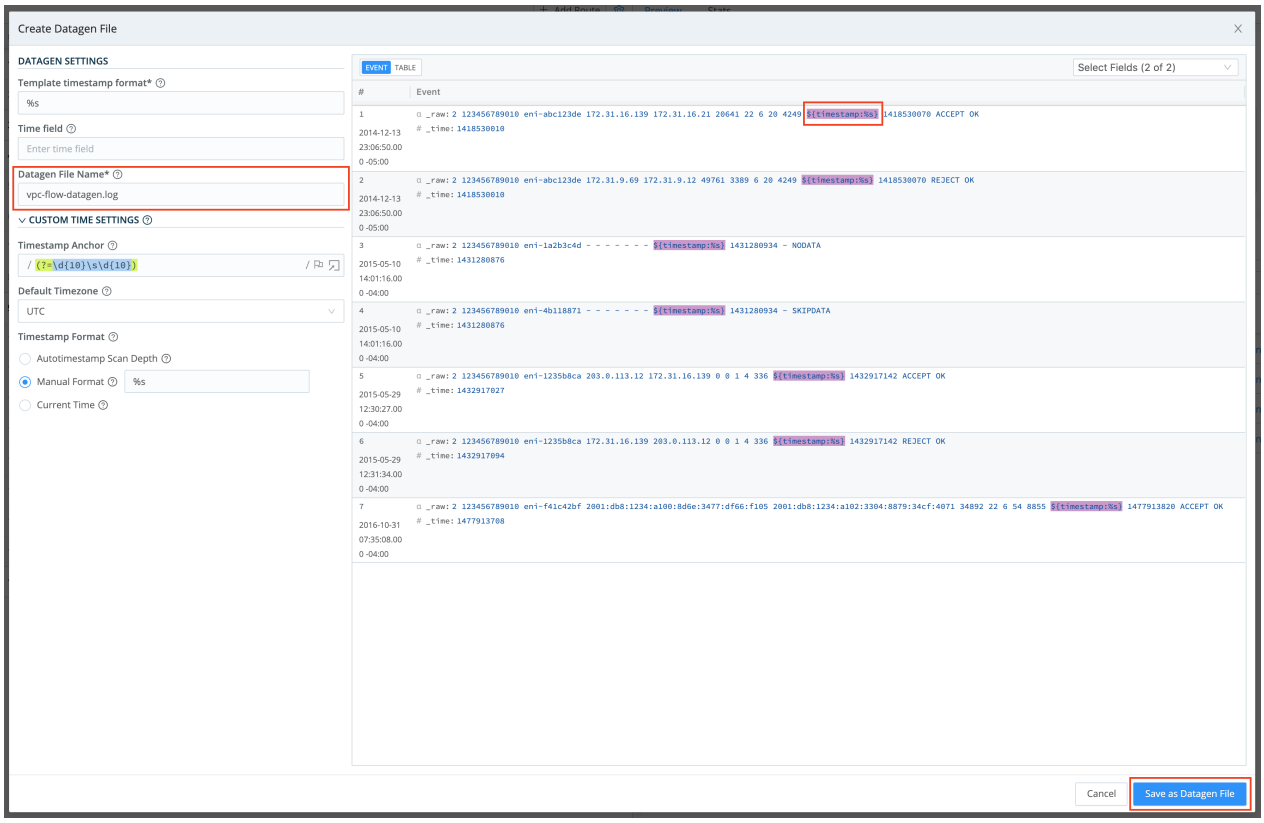


Creating a datagen template

On the resulting **Create Datagen File** screen:

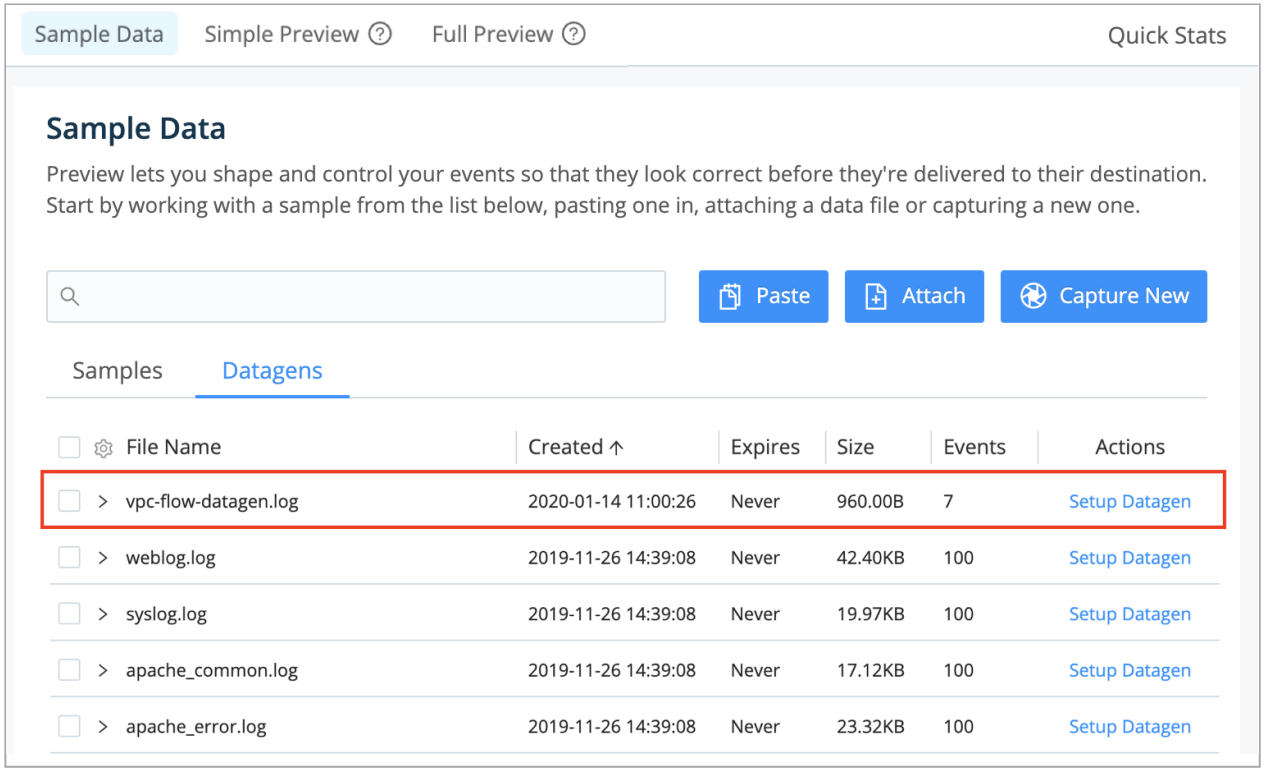
- Enter a file name, e.g.: `vpc-flow-datagen.log`
- Ensure that the timestamp template format is correct: `${timestamp: %s}`
  `${timestamp: <format>}` is a template that the datagen engine uses to insert the current time – in each newly generated event – using the given format. In this case, `%s` is the desired `strftime` format for the timestamp (i.e., the epoch).

Once you've verified these results, click **Save as Datagen File**.



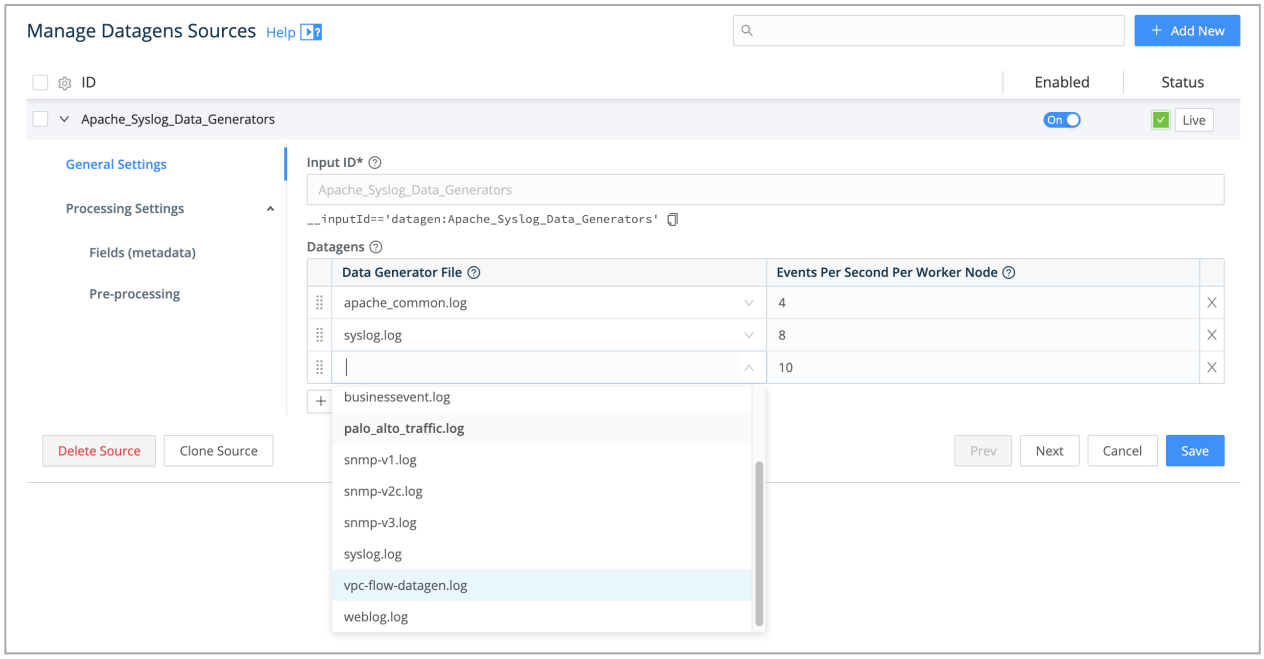
Saving a named datagen template

To confirm that the datagen file has been created, check **Preview** > **Datagens**.



Verifying datagen file creation

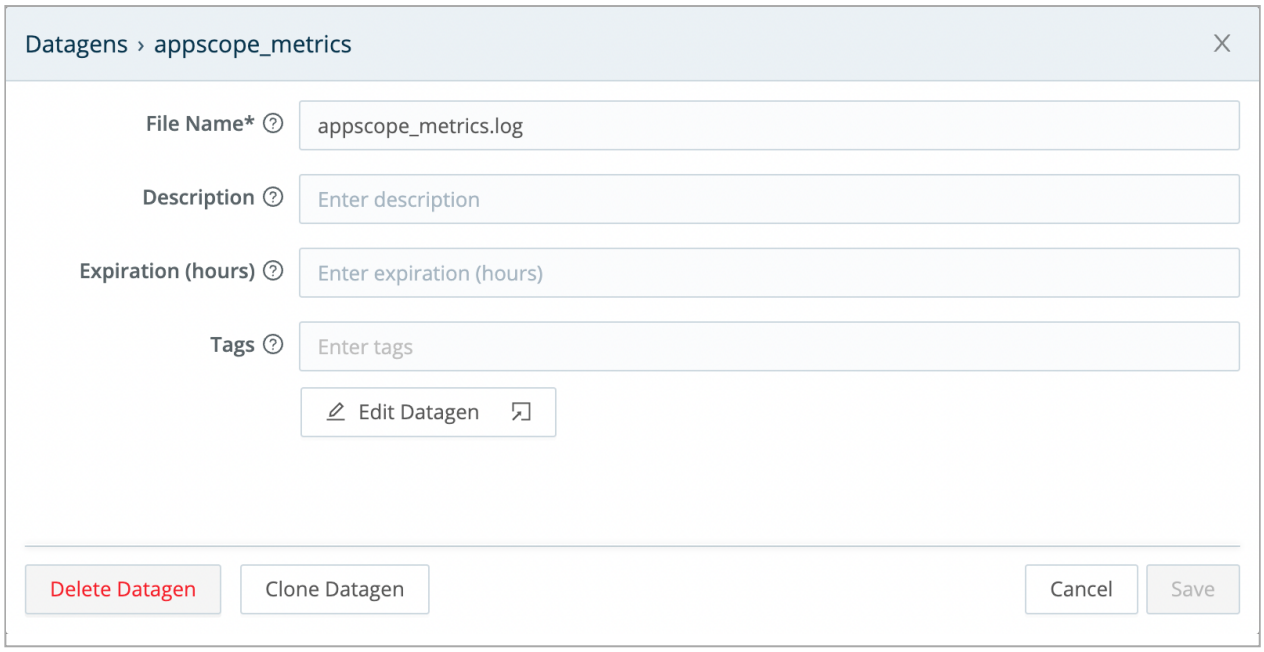
Now, to start using your newly created datagen file, go back to **Sources** > **Datagens**. Add it using the drop-down shown below.



Adding new template file to datagens Source

Modifying 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. Click 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, click 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. Click **Update Datagen** to resave the modified datagen, or click **Save as New Datagen** to give the modified version a new name.

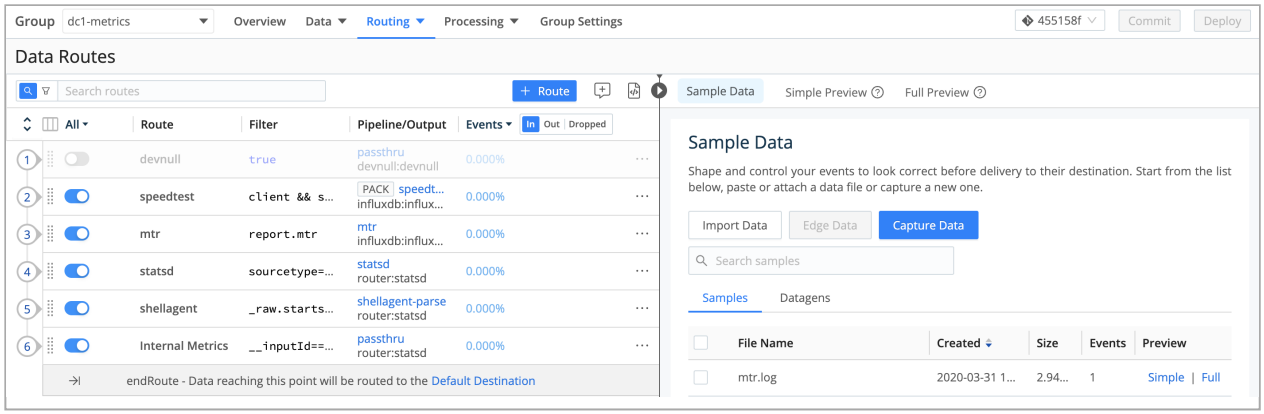
Editing a datagen

11.7. 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.

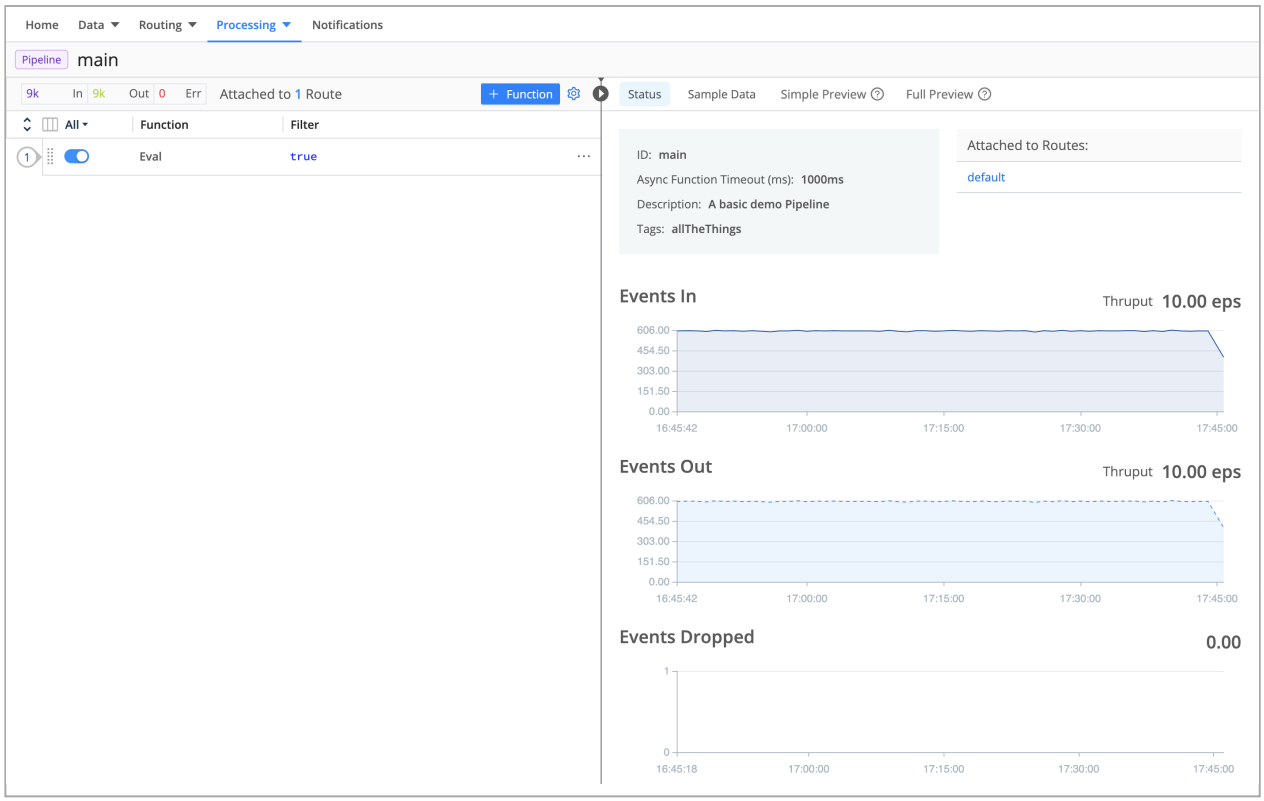


Preview options

> You can press the] (right-bracket) shortcut key to toggle the visibility of the Preview pane. (This shortcut works when no field has focus.)

Checking Pipeline Status

From the Pipelines page, click any Pipeline at left to display a **Status** tab in the right Preview pane. Click 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

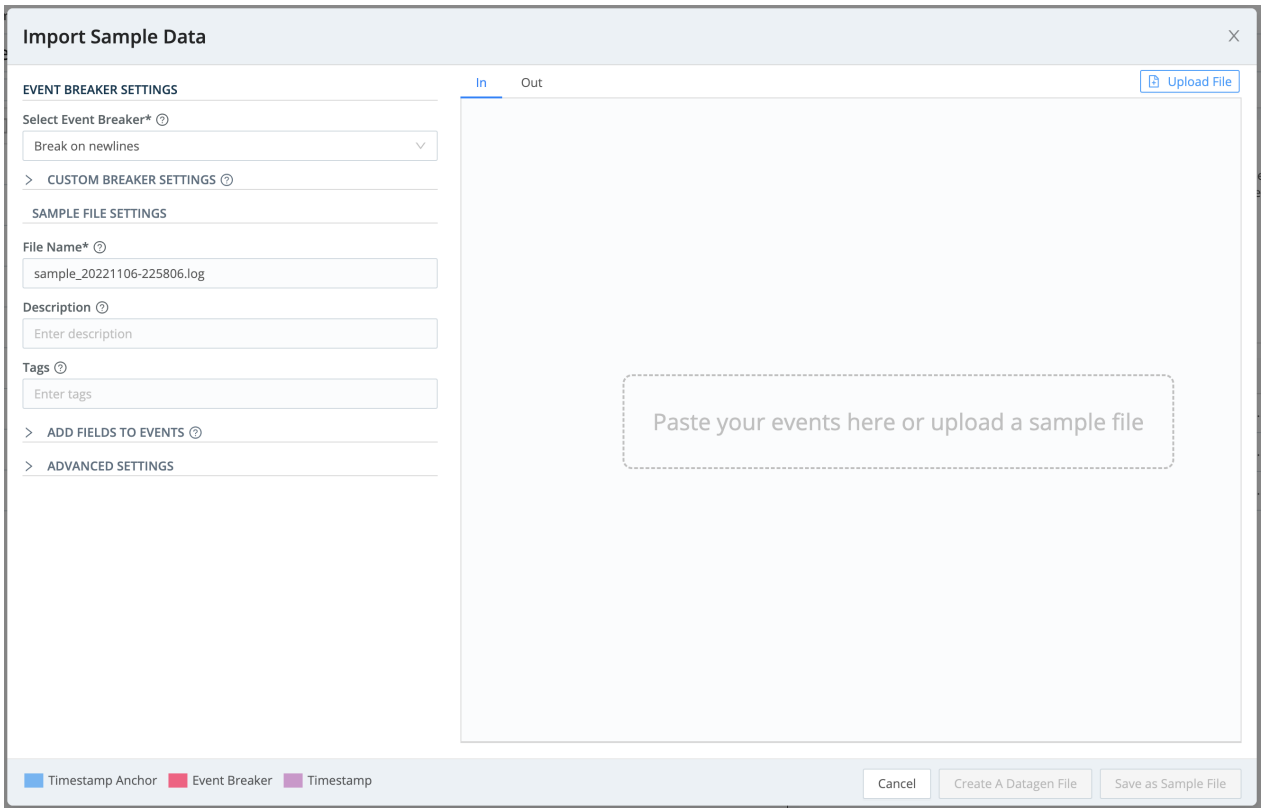
Adding Sample Data

When you're on the Pipelines or Routes page, you can add samples through any of the Preview pane's supported options: **Import Data** (which opens a modal where you can paste data from your clipboard or use the **Upload File** button); **Edge Data** (where you can import data from any Edge Nodes deployed on your account); and **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.

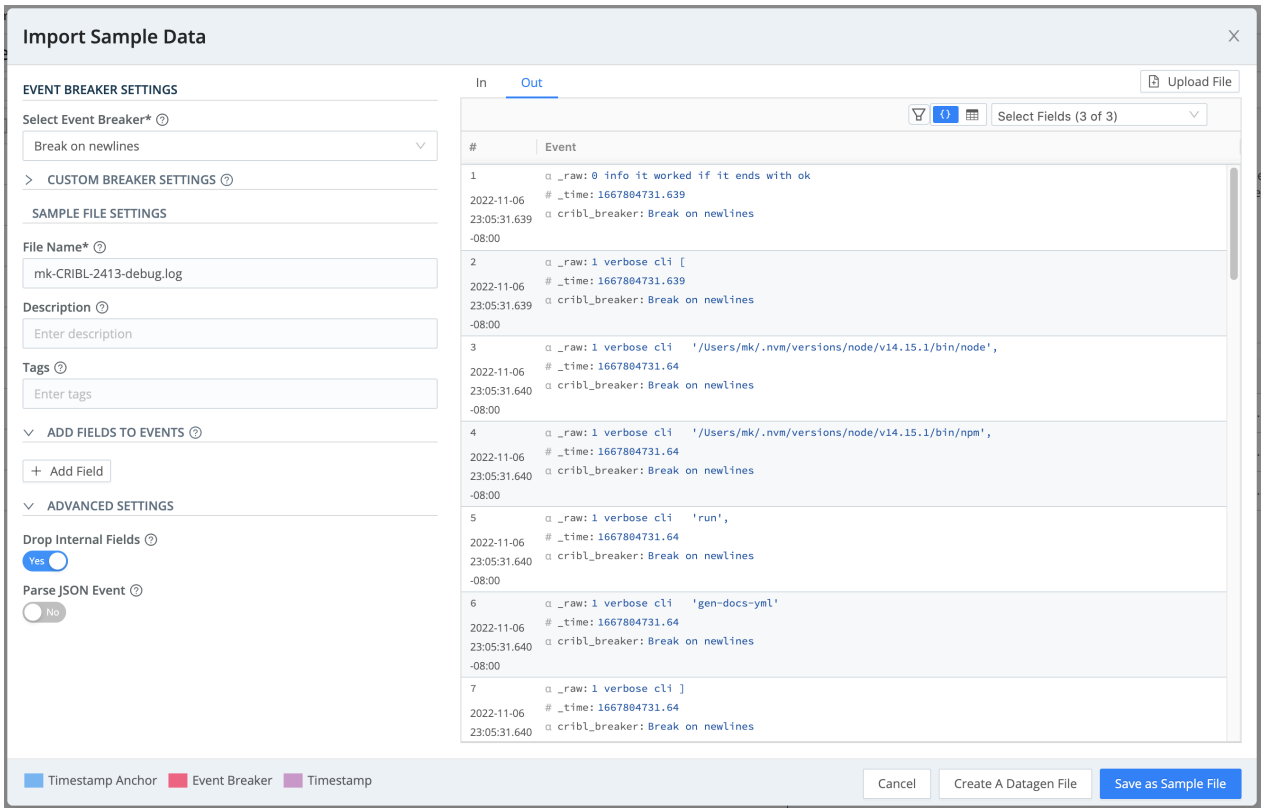
Paste Area

Once you've clicked the **Import Data** button, 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.

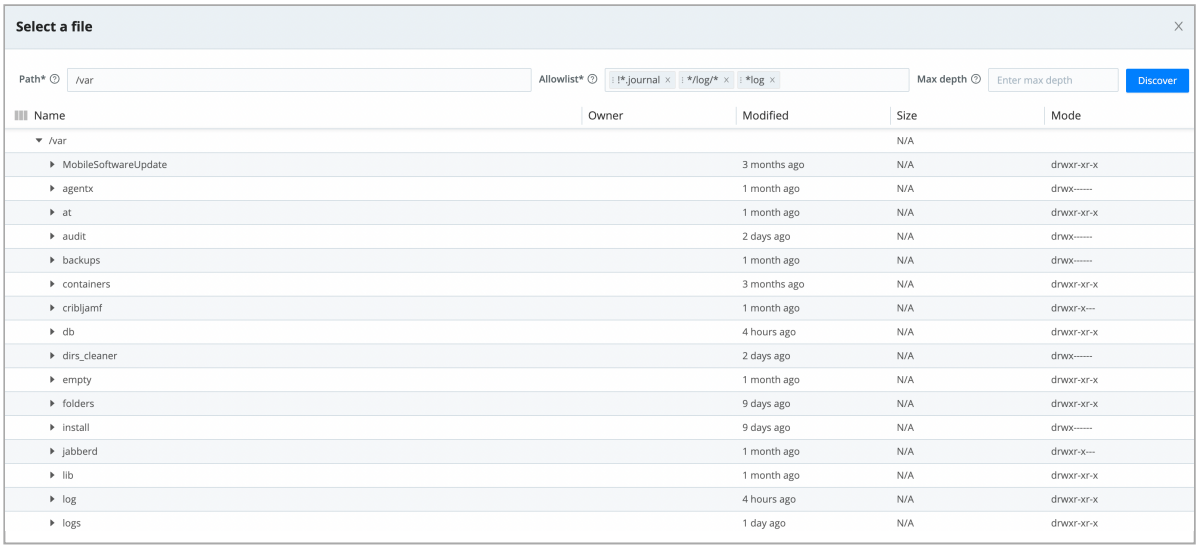


Edge Data

To upload data from a file on an Edge node:

1. Click the **Edge Data** button, and navigate to the Edge node where the file is stored.

   Clicking the Edge Node opens the **Select a file** modal, shown below.



Select a file modal

2. 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, click its name to add its contents to the **Add Sample Data** modal, where you'll finish configuring the data sample.

Event Breaker Settings

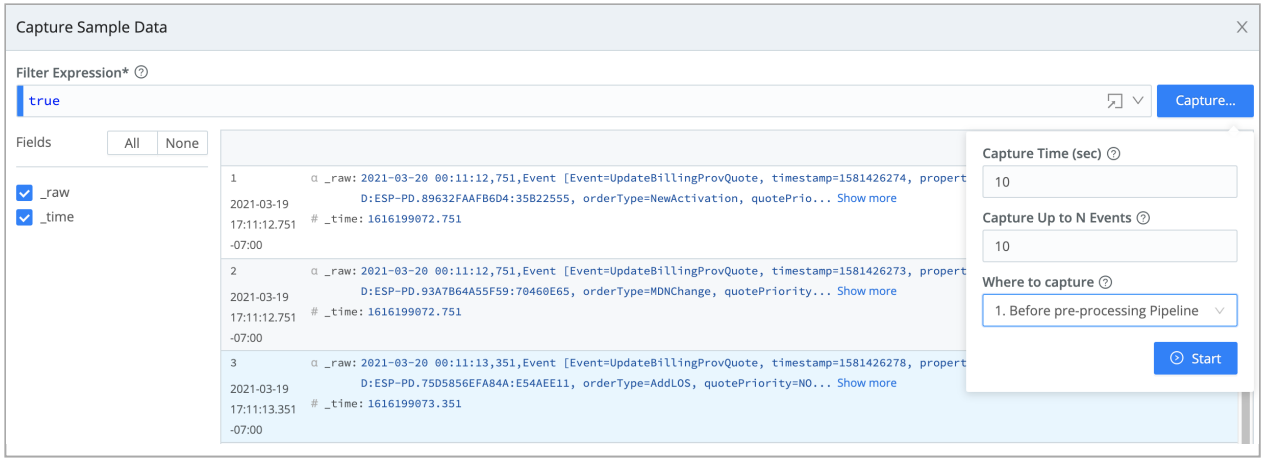
Cribl Event Breakers are regular expressions that tell Cribl Edge how to break the file or pasted content into events. Breaking will occur at the **start** of the match. Cribl Edge ships with several common breaker patterns out of the box, but you can also configure custom breakers. The UI here is interactive, and you can iterate until you find the exact pattern.

Troubleshooting 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.

Capturing Sample Data

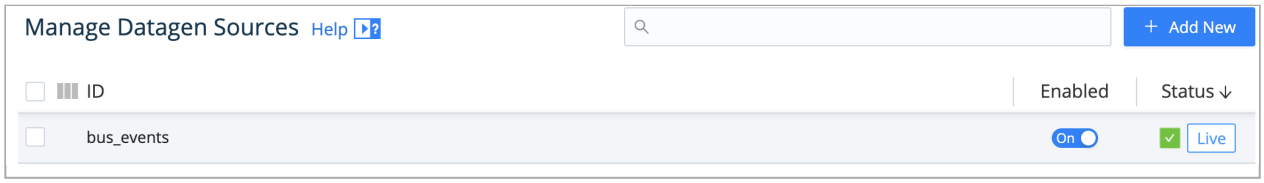
The **Capture Data** button opens a slightly different modal – it does not require event breaking. In the composite screenshot below, we've already captured some events using the **Capture** drop-down.



Capture Data > Capture Sample Data modal

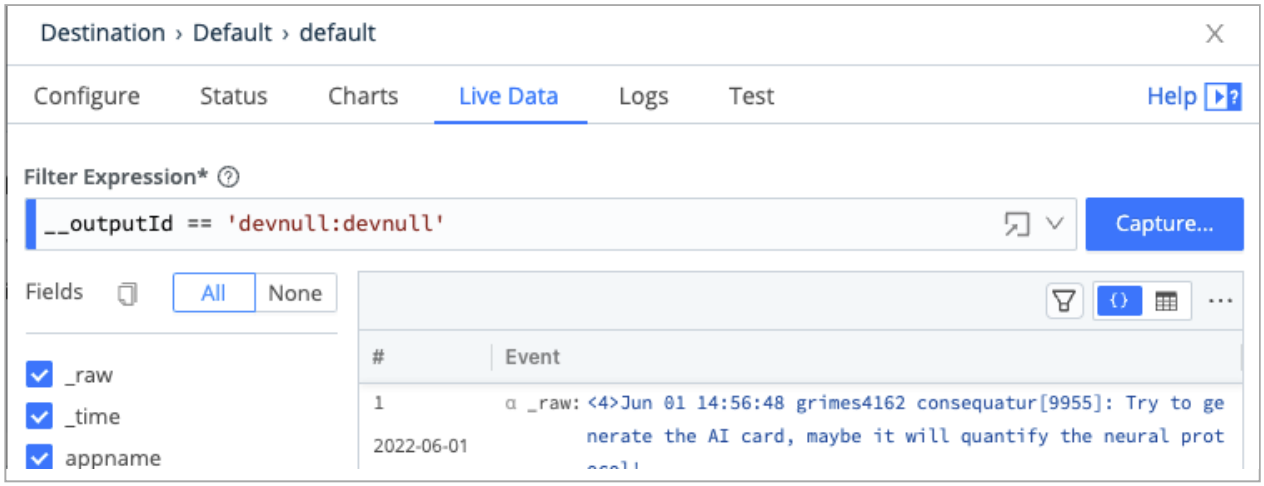
Capturing from a Single Source or Destination

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 clicking the **Live** button on the Source's or Destination's configuration row.



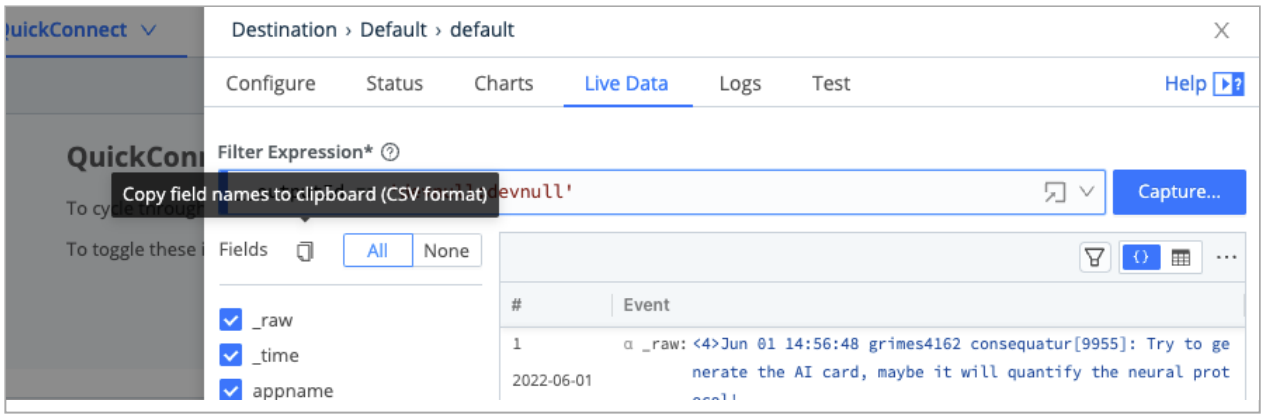
Source > Live button

You can similarly start an immediate capture from within an enabled Source's or Destination's configuration modal, by clicking 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

Controlling 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 the Leader at **Settings** > **Global Settings** > **General Settings** > **Limits** > **Max sample size**.
- On a single instance at **Settings** > **General Settings > Limits** > **Max sample size**.

- On distributed Worker Groups at `<group-name>` > **Fleet Settings** > **General Settings** > **Limits** > **Max sample size**.

In each location, the default limit is `256 KB`, and you can adjust this upward 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
#	numeric
b	boolean
m	metric
{}	JSON object
[]	array

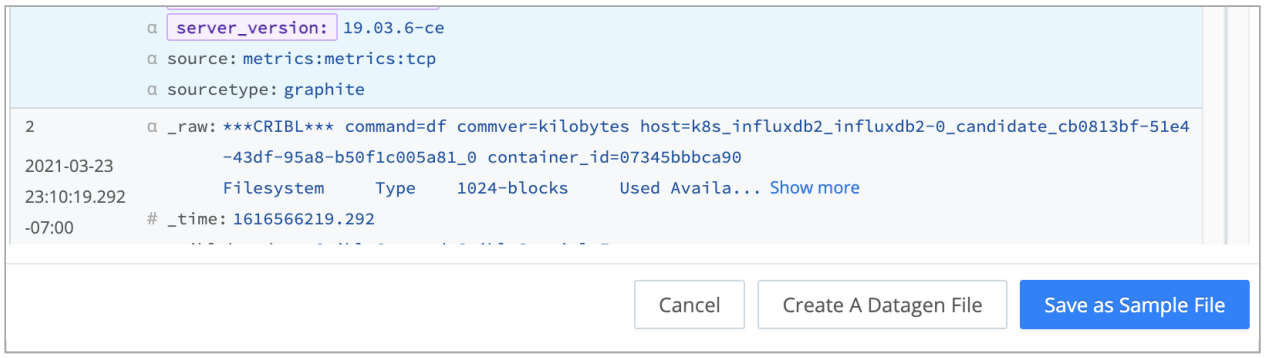
On JSON objects and arrays, you'll also see:

SYMBOL	MEANING
+	expandable

SYMBOL	MEANING
-	collapsible

Saving Sample Data

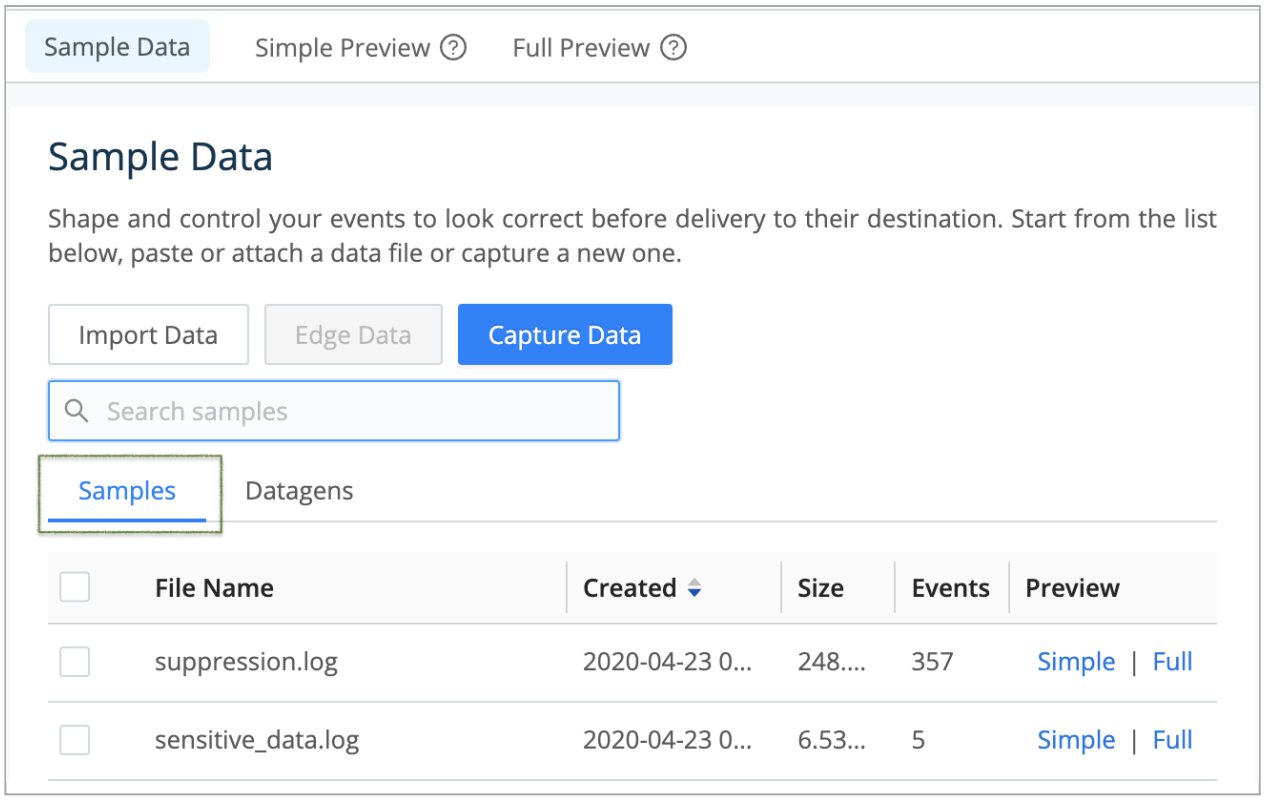
The Preview pane's **Import Data** or **Capture Data** modal, once you've successfully populated it with data, provides options to save the data as a sample and/or datagen file. Click the appropriate button, accept or modify the default/generated file name and other options, and confirm the save.



Saving sample data

Accessing and Managing Data Files

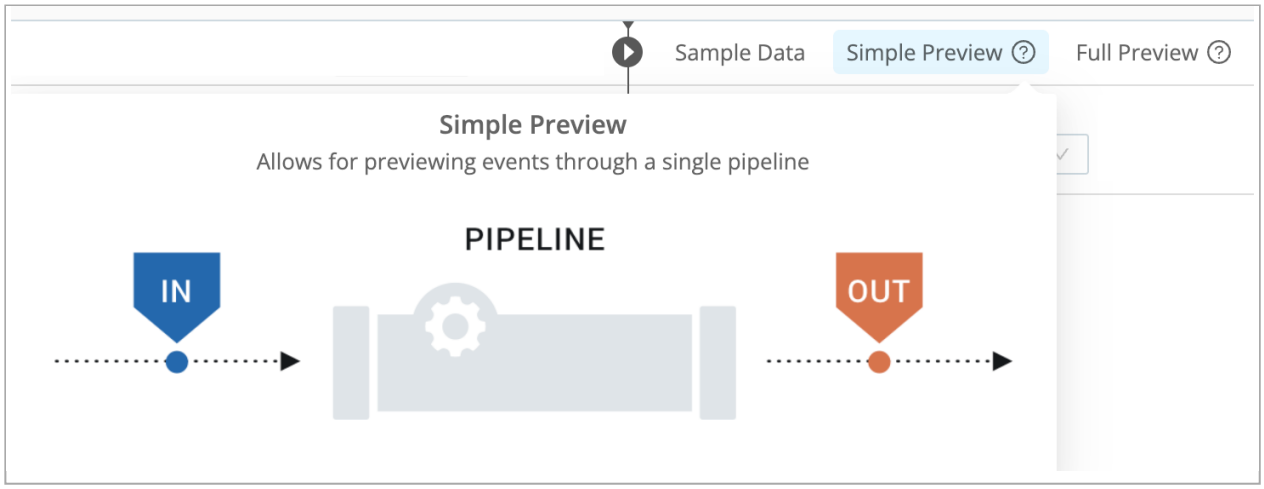
As you add more samples to your system, you can easily access them via the **Sample data file** drop-down. You can also manage and modify sample files via the **Samples** tab highlighted below.



Managing sample files

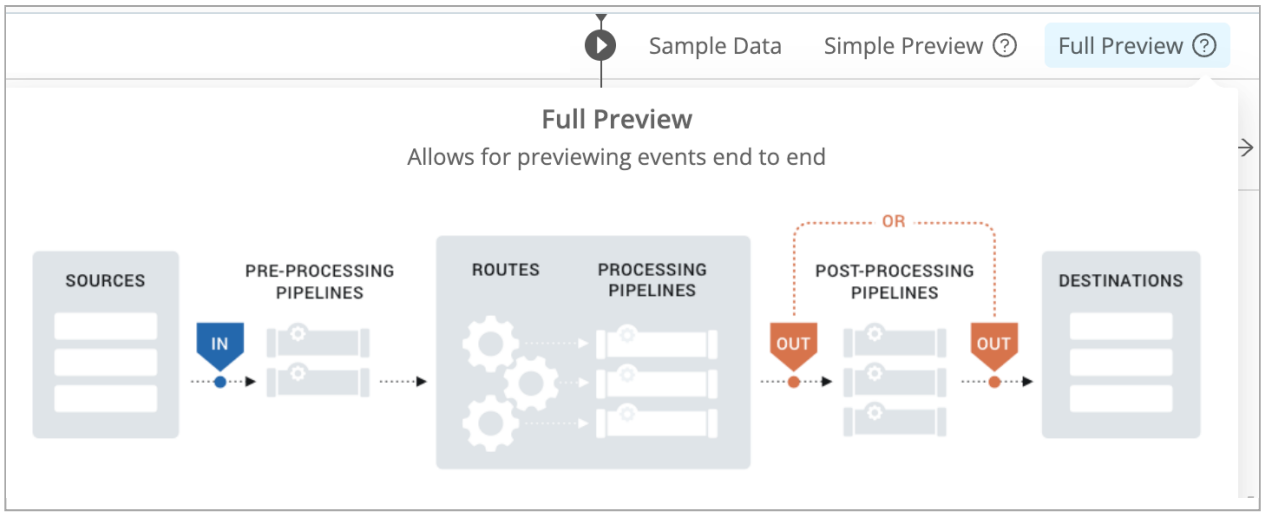
Simple Versus Full Preview

Click **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

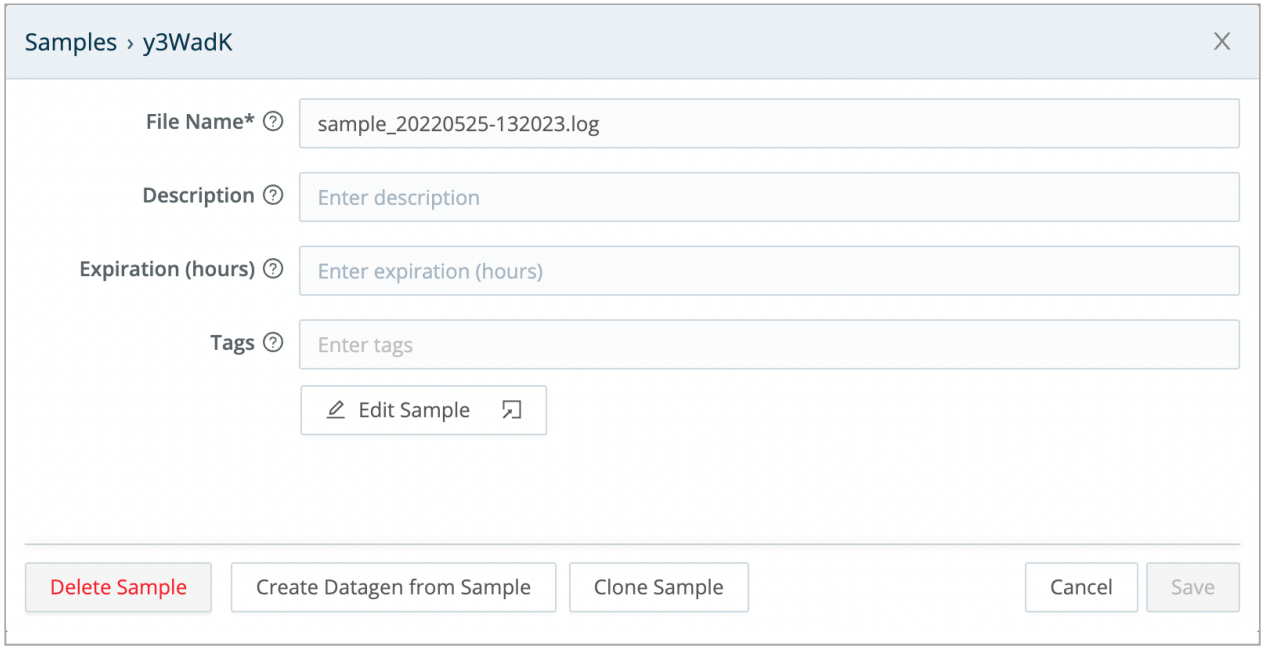
The **Full Preview** option gives you a choice of viewing events on the **OUT** side of either the processing or post-processing Pipeline. Selecting this option expands the Preview pane's upper controls to include an **Exit Point** drop-down, where you make this choice.



Full Preview schematic

Modifying 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. Click 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

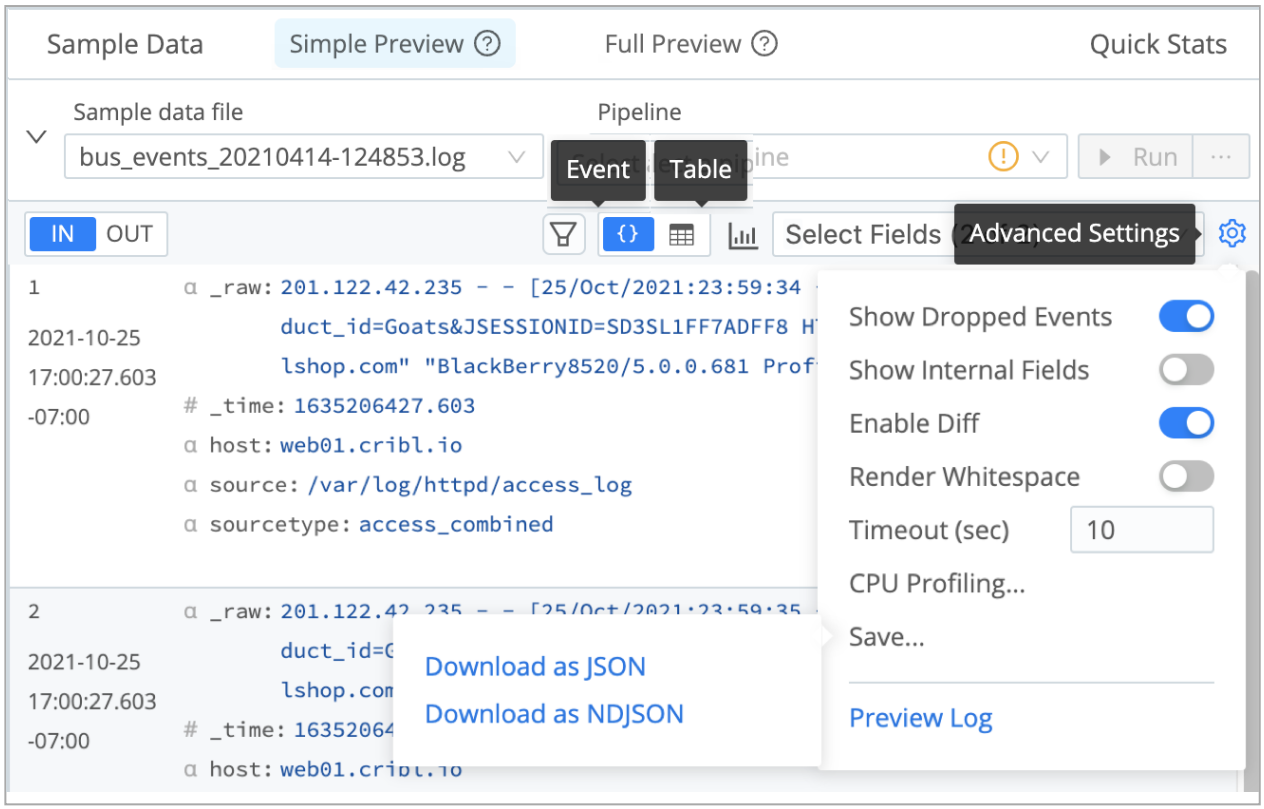
4. To make changes to the sample, click the modal's **Edit Sample** button. This opens the **Edit Sample** modal shown below, exposing the sample's raw data.

5. Edit the raw data as desired.
6. Click **Update Sample** to resave the modified sample, or click **Save as New Sample** to give the modified version a new name.

Editing a sample file

IN Tab: Displaying 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

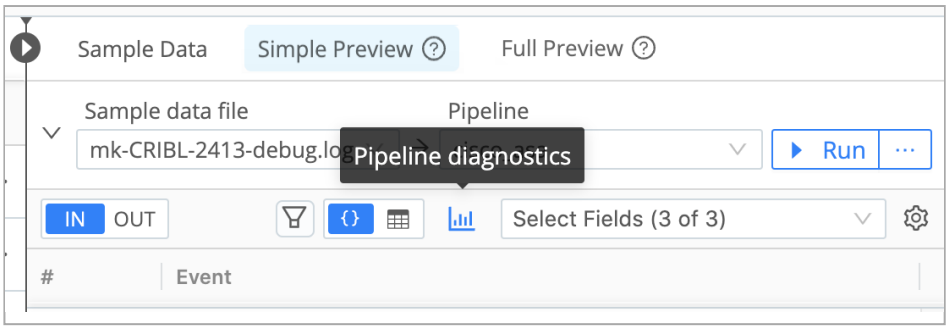
This toggles between displaying carriage returns, newlines, tabs, and spaces as white space, versus as(respectively) the symbols , , →, and ·.

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.

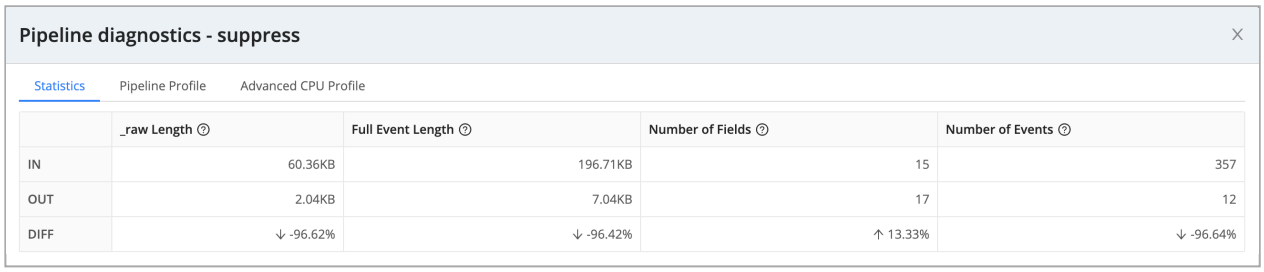
Pipeline Profiling

At the Preview pane's top, you can select a sample data file and Pipeline, then click 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.



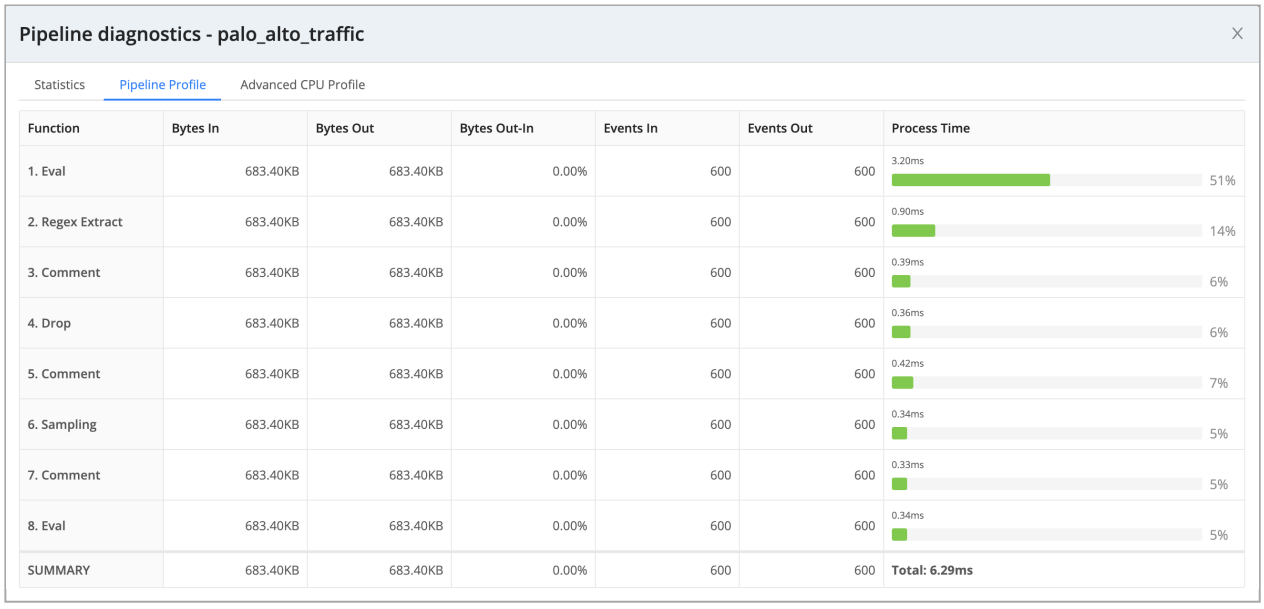
Pipeline diagnostics - suppress

Statistics | Pipeline Profile | Advanced CPU Profile

	_raw Length	Full Event Length	Number of Fields	Number of Events
IN	60.36KB	196.71KB	15	357
OUT	2.04KB	7.04KB	17	12
DIFF	↓ -96.62%	↓ -96.42%	↑ 13.33%	↓ -96.64%

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.



Pipeline diagnostics - palo_alto_traffic

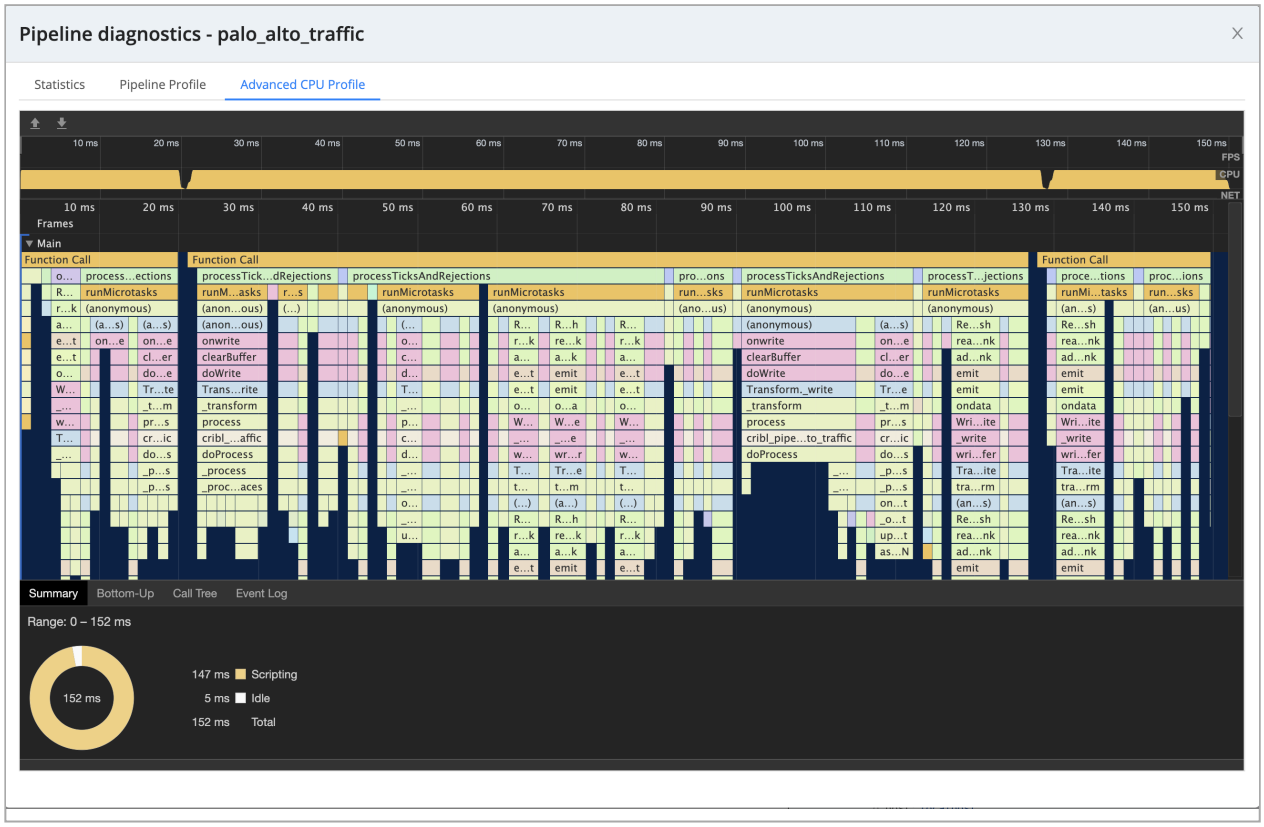
Statistics | Pipeline Profile | Advanced CPU Profile

Function	Bytes In	Bytes Out	Bytes Out-In	Events In	Events Out	Process Time
1. Eval	683.40KB	683.40KB	0.00%	600	600	3.20ms 51%
2. Regex Extract	683.40KB	683.40KB	0.00%	600	600	0.90ms 14%
3. Comment	683.40KB	683.40KB	0.00%	600	600	0.39ms 6%
4. Drop	683.40KB	683.40KB	0.00%	600	600	0.36ms 6%
5. Comment	683.40KB	683.40KB	0.00%	600	600	0.42ms 7%
6. Sampling	683.40KB	683.40KB	0.00%	600	600	0.34ms 5%
7. Comment	683.40KB	683.40KB	0.00%	600	600	0.33ms 5%
8. Eval	683.40KB	683.40KB	0.00%	600	600	0.34ms 5%
SUMMARY	683.40KB	683.40KB	0.00%	600	600	Total: 6.29ms

Pipeline diagnostics > Pipeline Profile tab

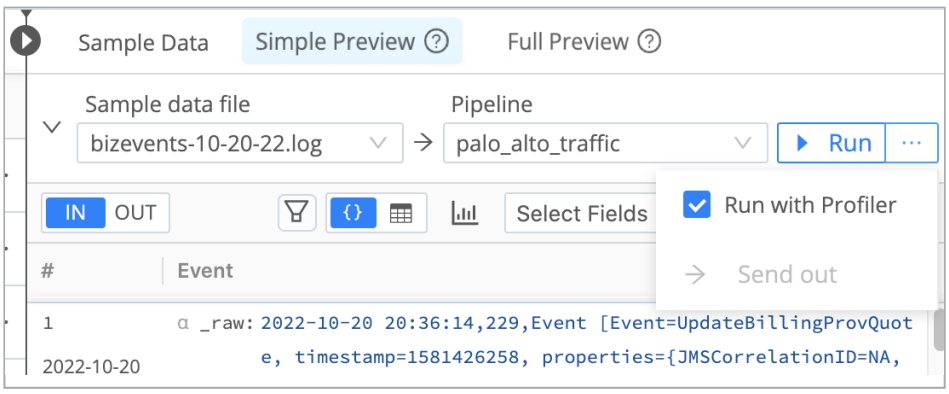
When you profile a Chain Function, this tab condenses all of the chained Pipeline's processing into a single **Chain** row. To see individual statistics on the aggregated Functions, profile the chained Pipeline separately.

The **Advanced CPU Profile** tab, available from the **Simple Preview** tab, displays richer details on individual Function stacks.



Pipeline diagnostics > Advanced CPU Profile tab

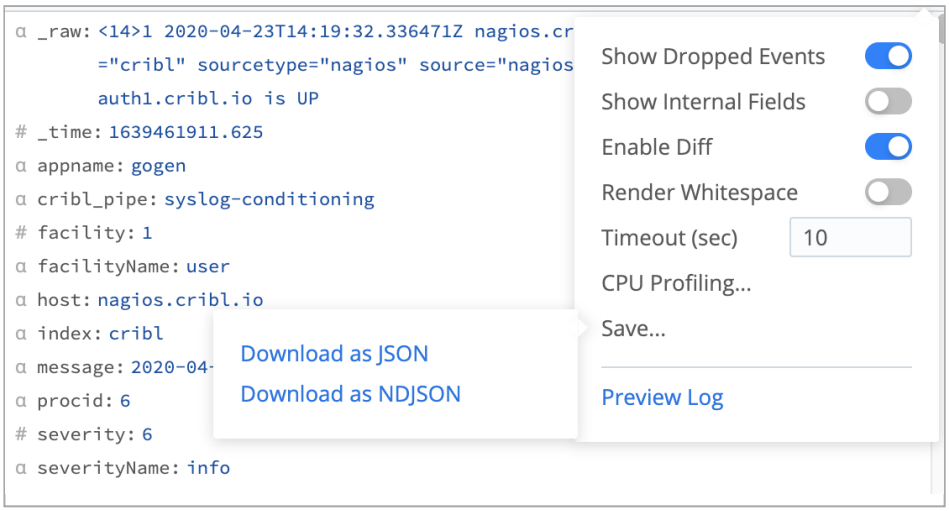
You can disable the profiling features – and regain their own (minimal) resources – by clicking the drop-down list beside the **Run** button and clearing the **Run with Profiler** check box.



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 **Downoad as NDJSON** (Newline-Delimied JSON) option.



Saving sample data as JSON

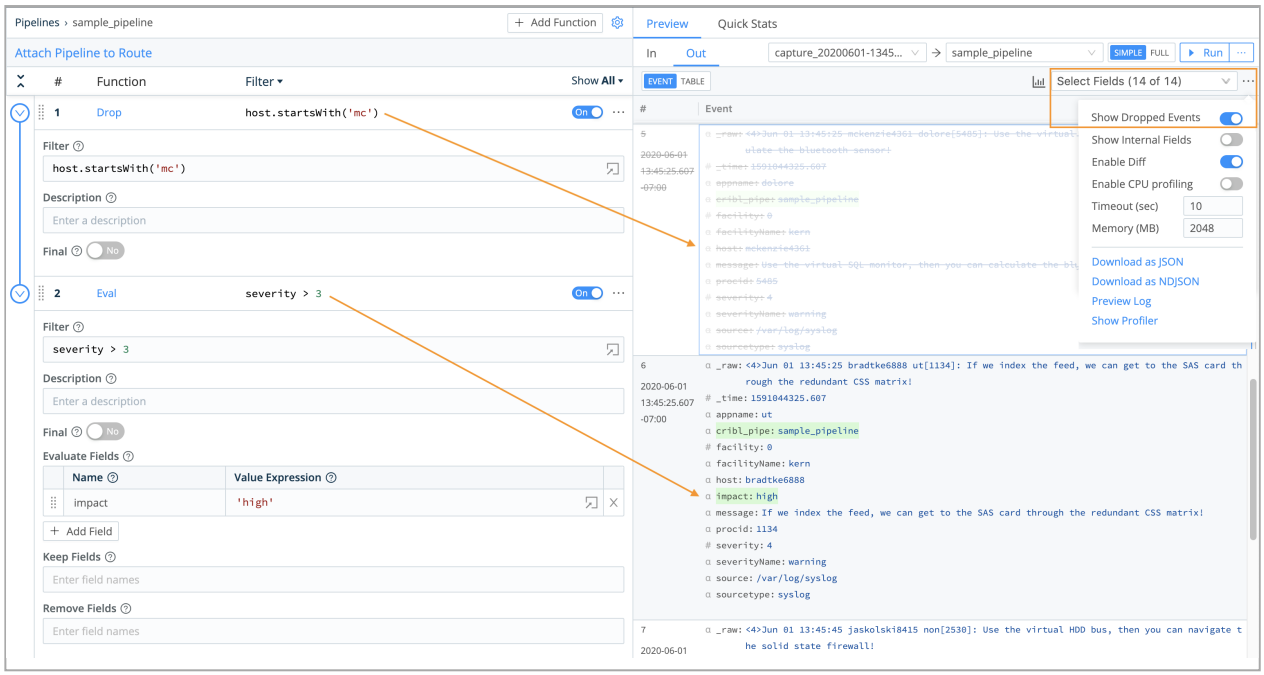
Preview Log

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: Displaying 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** slider.
- **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.



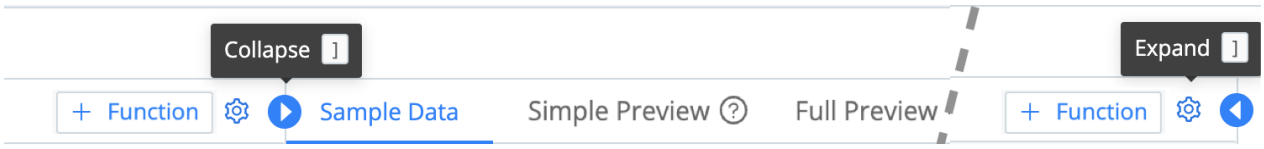
Dropped and added fields in a Pipeline's output

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.

Managing the Preview Pane

With the Routes or Pipelines page displayed in the left pane, hover over the pane divider (in the headers row) to display the Collapse/Expand toggle shown in the composite screenshot below.



Collapse / Expand toggle (composite)

Click 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.)

Click Expand at your browser's right edge to restore the split view. The pane divider will snap back to wherever you last dragged it.

11.8. 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

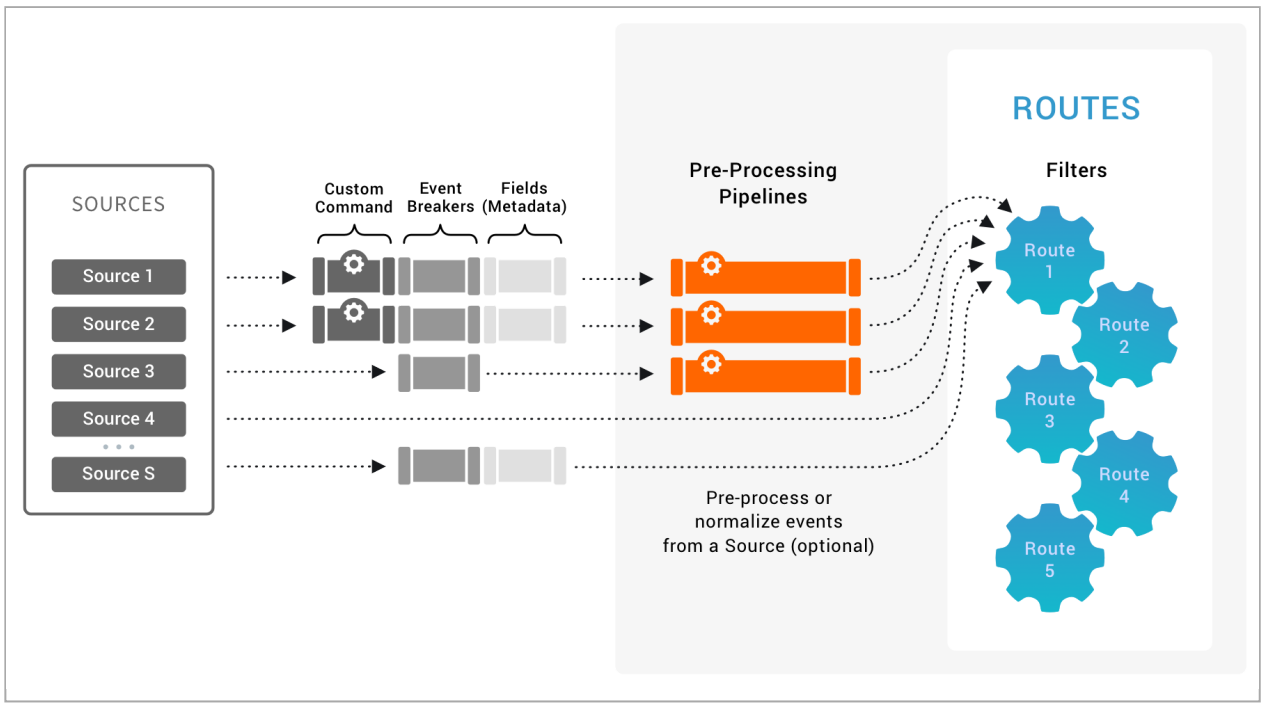
Use a sample of your real data in Data Preview. Sample data can come from a sample Source file that you upload or paste into Cribl Edge.

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.

12. Functions

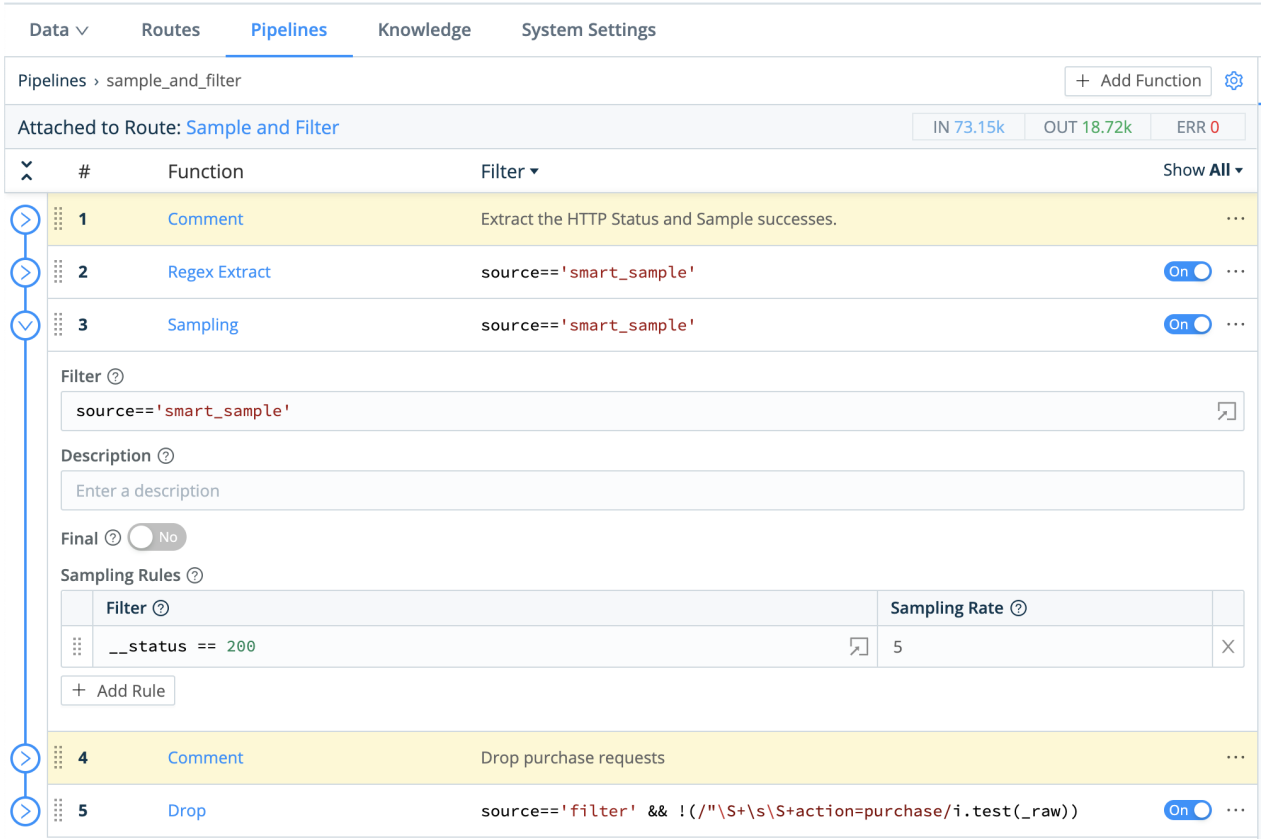
When events enter a Pipeline, they're processed by a series of Functions. At its core, a Function is code that executes on an event, and it encapsulates the smallest amount of processing that can happen to that event.

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 turn Functions **On**/**Off** within a Pipeline as necessary. This enables you to 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

Similar to the `Final` toggle in Routes, the `Final` toggle here controls the flow of events at the Function level. Its states are:

- `No` (**default**): means that matching events processed by this Function will be passed down to the next Function.
- `Yes` : means that this Function is the last one that will be applied to matching events. All Functions further down the Pipeline will be skipped. A Function with `Final` set to `Yes` will display an **F** indicator in the Pipeline stack.

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

- 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)
- 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

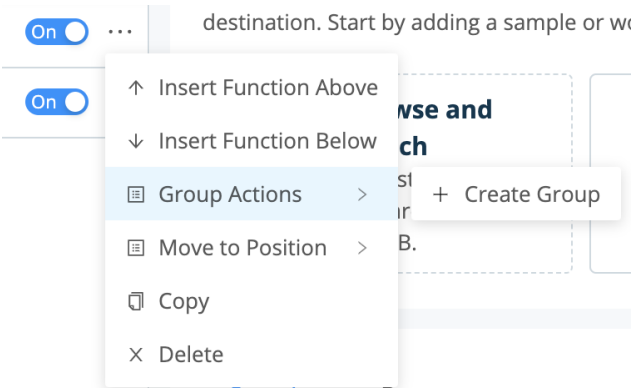
> For more usage examples, download Cribl's Stream Cheat Sheet (Quick Reference Card).

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.

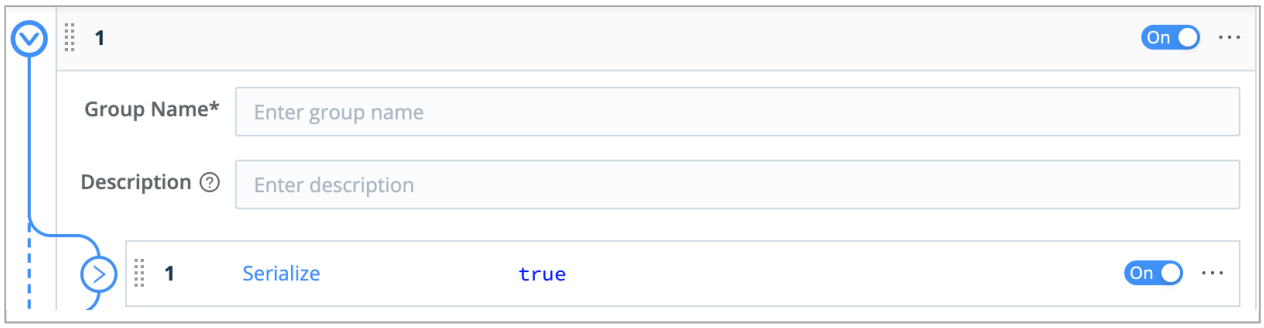
> Function groups work much like Route groups.

To build a group from any Function, click the Function's ••• (Options) menu, then select **Group Actions > Create Group**.



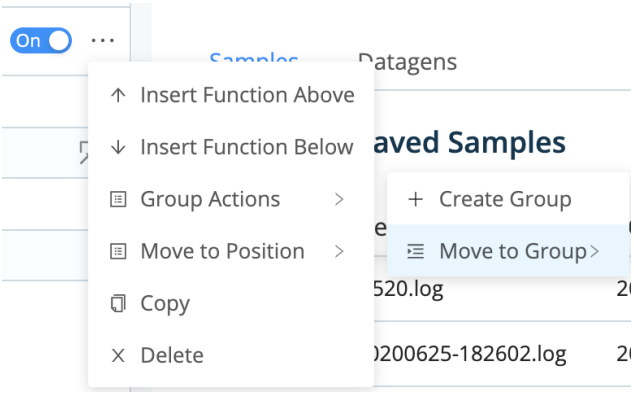
Creating a group

You'll need to enter a **Group Name** before you can save or resave the Pipeline. Optionally, enter a **Description**.



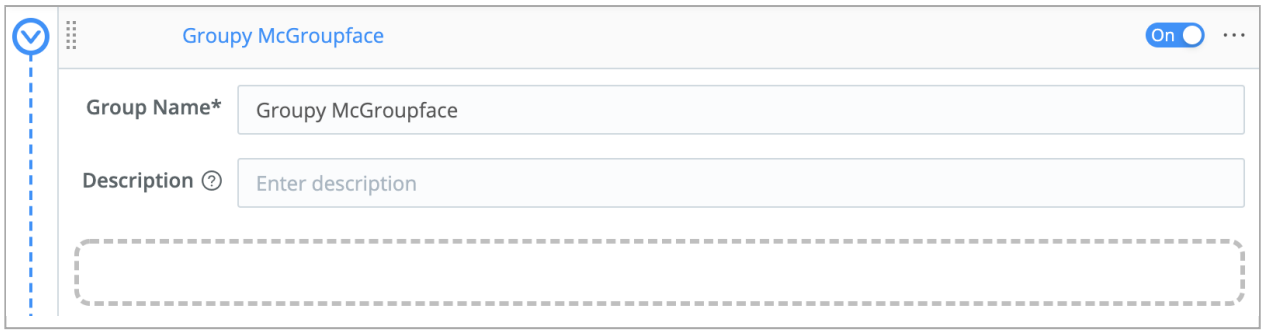
Naming a group

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**.



Expanded Group Actions submenu

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

12.1. 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
ntrusted,trusted,ethernet1/3,ethernet1/2,log-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
```

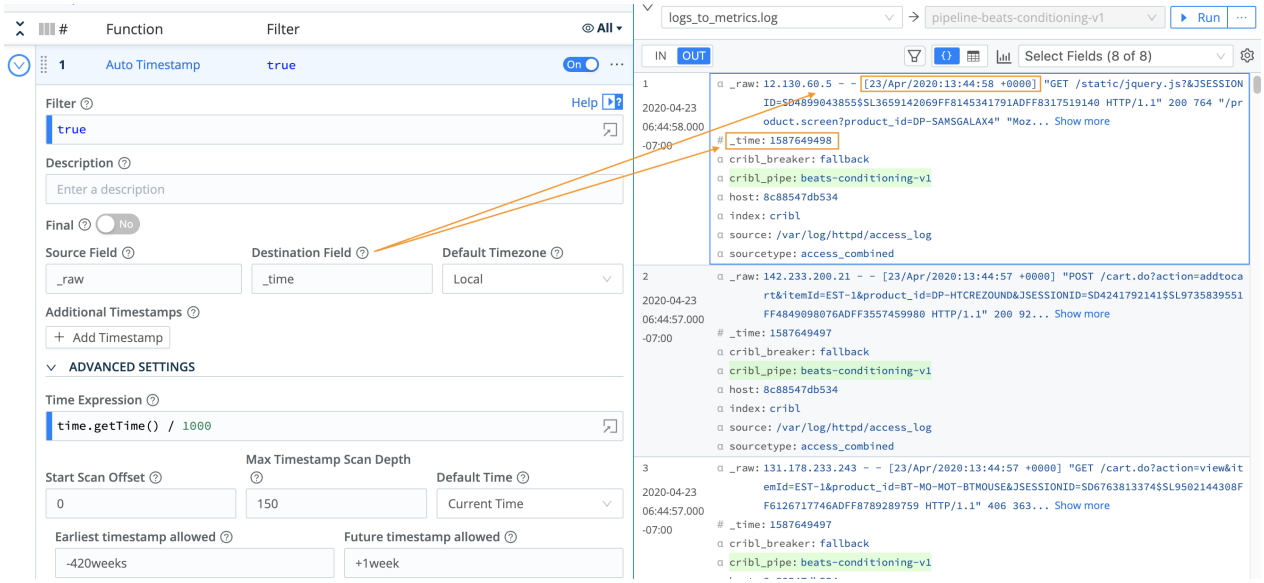
To add this sample (after creating an Auto Timestamp Function with the above **Filter** expression): Go to **Preview** > **Add a Sample** > **Paste a Sample**, and add the data snippet above. Do not make any changes to timestamping or line breaking, and select **Save as Sample File**.

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 1569006235`
- **GMT**: Friday, 20 September 2019, 7:03:55 PM GMT
- **Your Local Time**: Friday, 20 September 2019 PDT, 12:03:55 AM **GMT** -07:00

Because no explicit timezone has been set (under **Default Timezone**), `_time` will inherit the **Local** timezone, which in this example is `GMT -07:00`.



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.

12.2. 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: Aggregate 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.

- Note: When used without `as()`, the aggregate's output will be placed in a field labeled `<aggFunction>_<fieldName>`. If there are conflicts, the last aggregate wins. For example, given two aggregates – `sum(bytes).where(action=='REJECT')` and `sum(bytes)` – the latter one (`sum_bytes`) is the winner.

Group by Fields: Fields to group aggregates by. Supports wildcard expressions.

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.

List of Aggregate Functions

- `avg(expr:FieldExpression)`: Returns the average of the values of the parameter.
- `count(expr:FieldExpression)`: Returns the number of occurrences of the values of the parameter.
- `dc(expr: FieldExpression, errorRate: number = 0.01)`: Returns the estimated number of distinct values of the `<expr>` parameter, within a relative error rate.
- `distinct_count(expr: FieldExpression, errorRate: number = 0.01)`: Returns the estimated number of distinct values of the `<expr>` parameter, within a relative error rate.
- `earliest(expr:FieldExpression)`: Returns the earliest (based on `_time`) observed value of the parameter.
- `first(expr:FieldExpression)`: Returns the first observed value of the parameter.
- `histogram(expr:FieldExpression, buckets: number[])`. Returns the average of the values of the parameter and generates a field with the same name as the aggregate 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 the parameter.
    - `_count` – the number of values seen.
    - `_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:

  ```
  histogram (request_duration, [50, 100, 500])
  ```

- `last(expr:FieldExpression)`: Returns the last observed value of the parameter.
- `latest(expr:FieldExpression)`: Returns the latest (based on `_time`) observed value of the parameter.

- `list(expr:FieldExpression[, max:number, excludeNulls: boolean = true])`: Returns a list of values of the parameter.
    - 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 value of the parameter.
- `median(expr:FieldExpression)`: Returns the middle value of the sorted parameter.
- `min(expr:FieldExpression)`: Returns the minimum value of the parameter.
- `mode(expr: FieldExpression[, excludeNulls: boolean = true])`: Returns the single most frequently encountered value of the `<expr>` parameter.
    - Optional `excludeNulls` boolean excludes null and undefined values from results. If included, defaults to `true`.
- `per_second(expr:FieldExpression)`: Returns the per second rate (based on `_time`) observed value of the parameter.
- `perc(level: number, expr: FieldExpression)`: Returns `<level>` percentile value of the numeric values of the `<expr>` parameter.
- `rate(expr:FieldExpression, timeString: string = '1s')`: Returns the rate (based on `_time`) observed value of the parameter.
- `stdev(expr:FieldExpression)`: Returns the sample standard deviation of the values of the parameter.
- `stdevp(expr:FieldExpression)`: Returns the population standard deviation of the values of the parameter.
- `sum(expr:FieldExpression)`: Returns the sum of the values of the parameter.
- `summary(expr:FieldExpression)[, quantiles: number[]])`. Returns the average of the values of the parameter 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 values of the parameter.
  - `_count` – the number of values seen.
  - `_quantiles` – an object whose keys are the quantiles defined in the function, and whose values are the quantile value across the values of the parameter. 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:

```
summary (request_duration, [0.1, 0.5, 0.95])
```

- `sumsq(expr:FieldExpression)`: Returns the sum of squares of the values of the parameter.
- `top(expr: FieldExpression, count: number[, excludeNulls: boolean = true])`: Returns the most frequently encountered values of the `<expr>` parameter, 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, errorRate:number, excludeNulls: boolean = true])`: Returns a list of distinct values of the parameter.
  - Optional `max` parameter limits the number of values returned; if omitted, the default is `0`, meaning return all distinct values.
  - 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.
  - Optional `excludeNulls` boolean excludes null and undefined values from results. If included, defaults to `true`.
- `variance(expr:FieldExpression)`: Returns the sample variance of the values of the parameter.
- `variancep(expr:FieldExpression)`: Returns the population variance of the values of the parameter.

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.

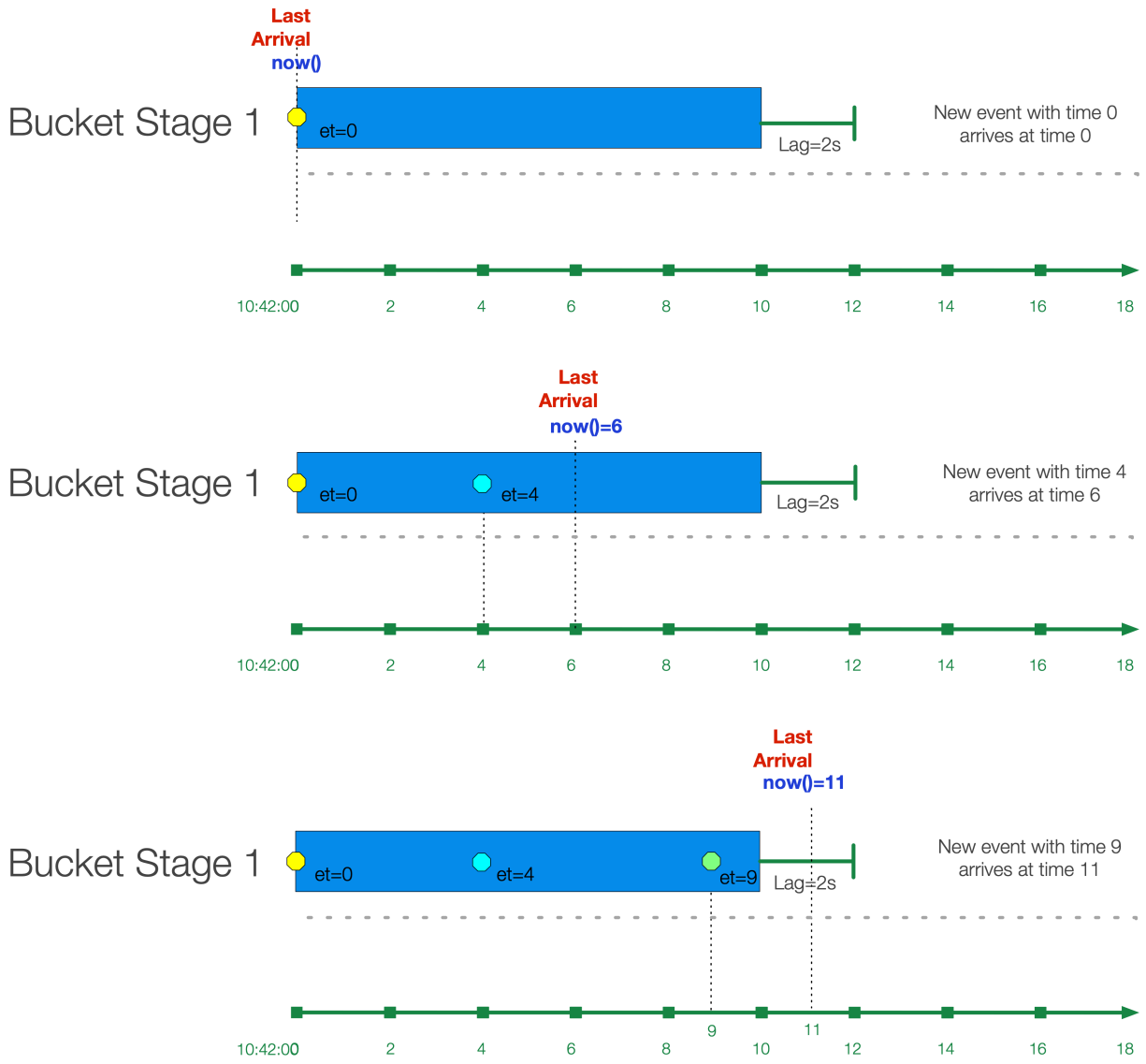
This is especially necessary for high-cardinality data. (Both settings default to unlimited, but we recommend setting defined limits based on testing.)

How Do Time Window Settings Work?

Lag Tolerance

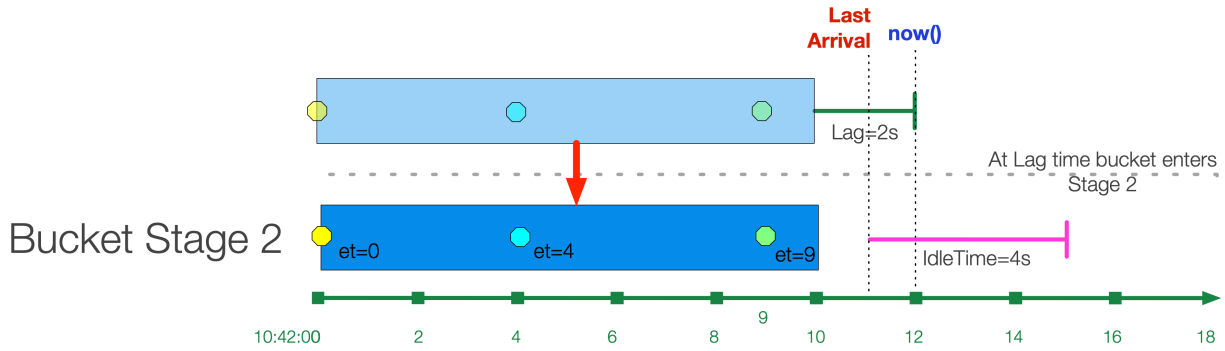
As events are aggregated into windows, there is a good chance that most will arrive later than their event time. For instance, given a `10s` window (`10:42:00 - 10:42:10`), an event with timestamp `10:42:03` might come in 2 seconds later at `10:42:05`.

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.

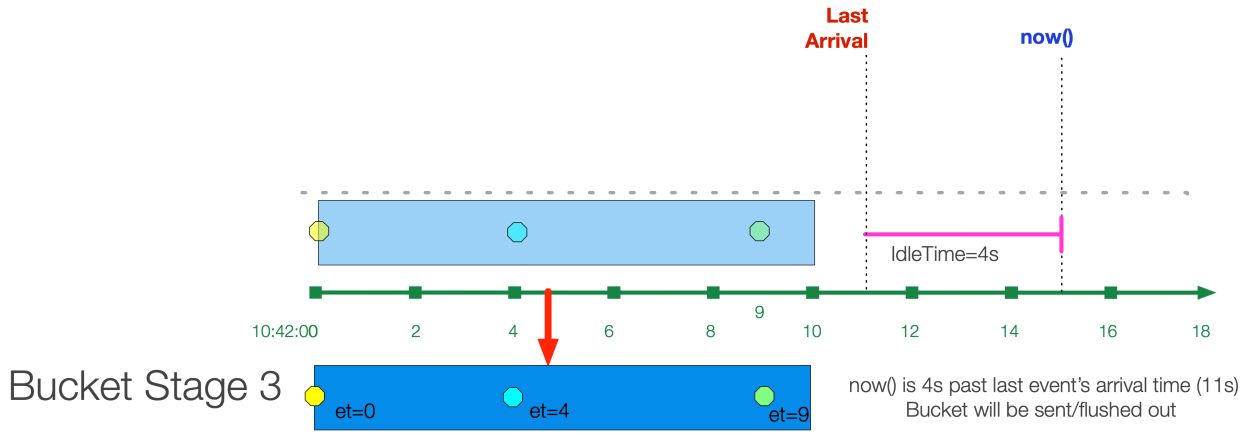
After the lag time expires, the bucket moves to Stage 2.



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.

12.3. 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
```

12.4. 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.

> This Function includes guardrails against circular references. Still, Cribl recommends that you keep chained configurations simple and understandable by all your users. Inserting a Chain Function currently imposes a performance hit of about 10%, compared to including all processing in the original Pipeline or Pack.

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.

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.

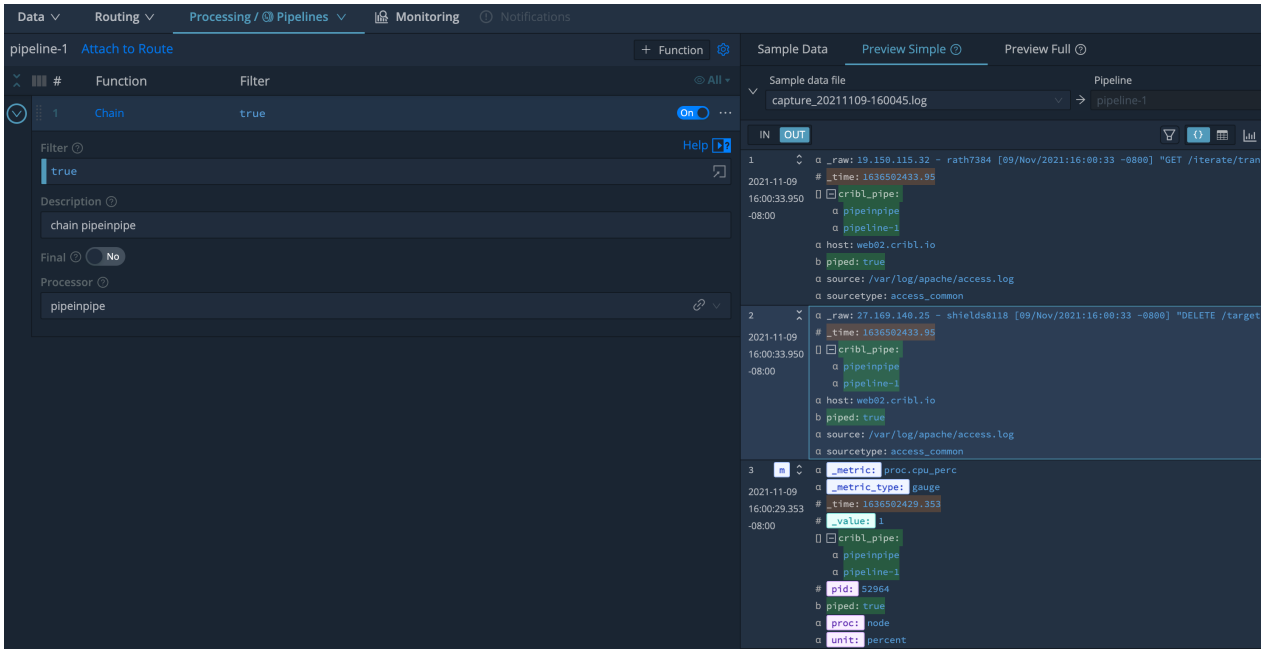
Description: Optionally, add a simple description of this Function's purpose. Defaults to empty.

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

12.5. Clone

The Clone Function clones events, with optional added fields. Cloned events will be sent 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.

12.6. 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`, and `window`.

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:

```
__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

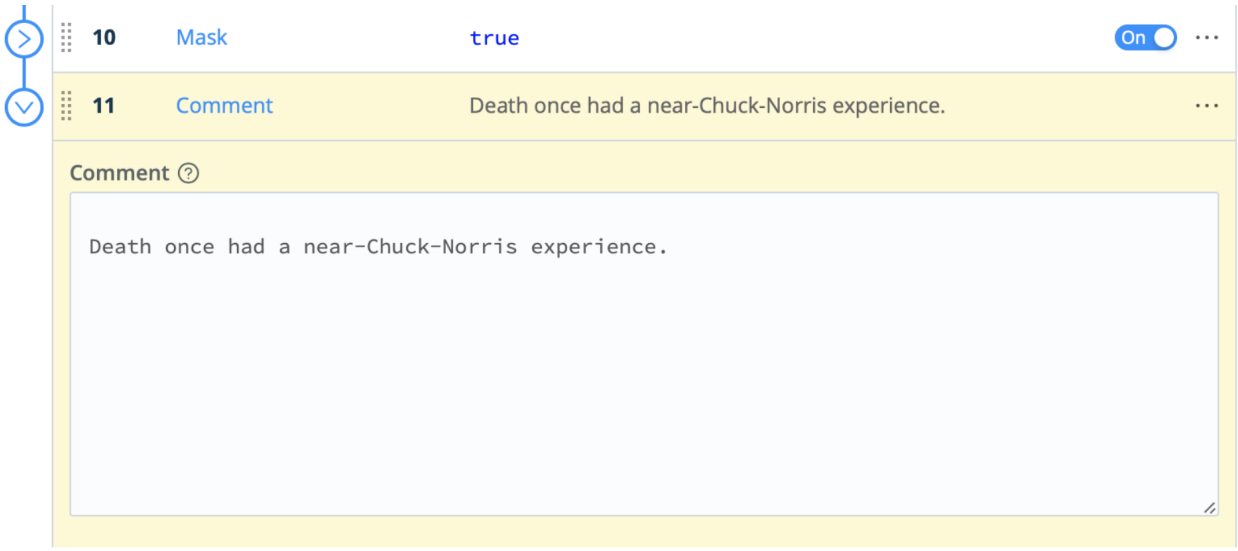
- Indexed collections
- Keyed collections
- Working with Objects

12.7. 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.

Usage

Comment: Add your comment as plain text in this field.



Examples

This comment labels the Pipeline's next function:

15
Eval
true
On
Filter
true
Description
Enter a description
Final
No
Evaluate Fields
Name
Value Expression
compromised
(C.Lookup('compromised-ips.csv').match(src_ip) || C.…
Add Field

12.8. 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

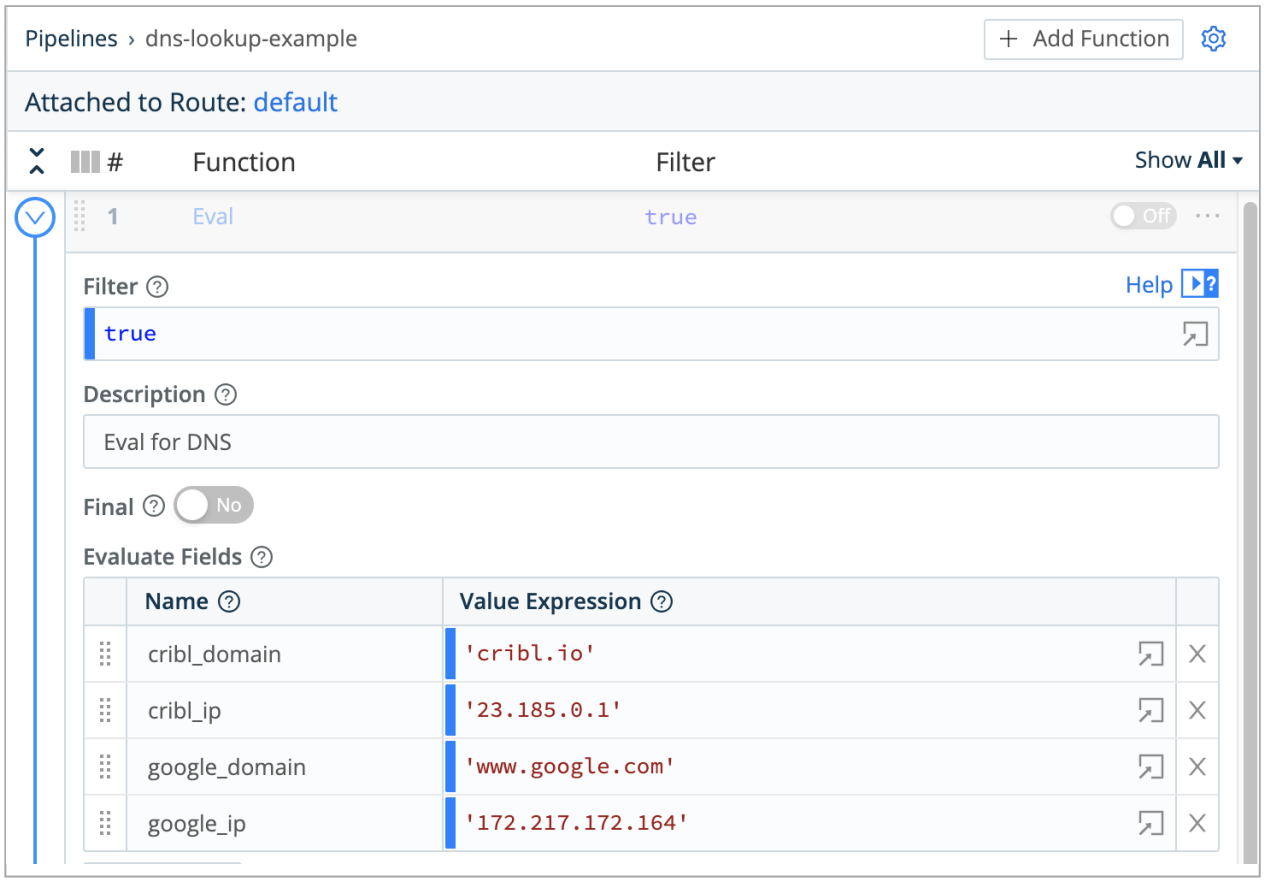
DNS server(s) overrides: IP address(es), in RFC 5952 format, of the DNS server(s) to use for resolution. IPv4 examples: `1.1.1.1`, `4.2.2.2:53`. IPv6 examples: `[2001:4860:4860::8888]`, `[2001:4860:4860::8888]:1053`. If this field is not specified, Cribl Edge will use the system's DNS server.

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`, contact Cribl Support to understand the implications. Highest allowed value is `10000`.

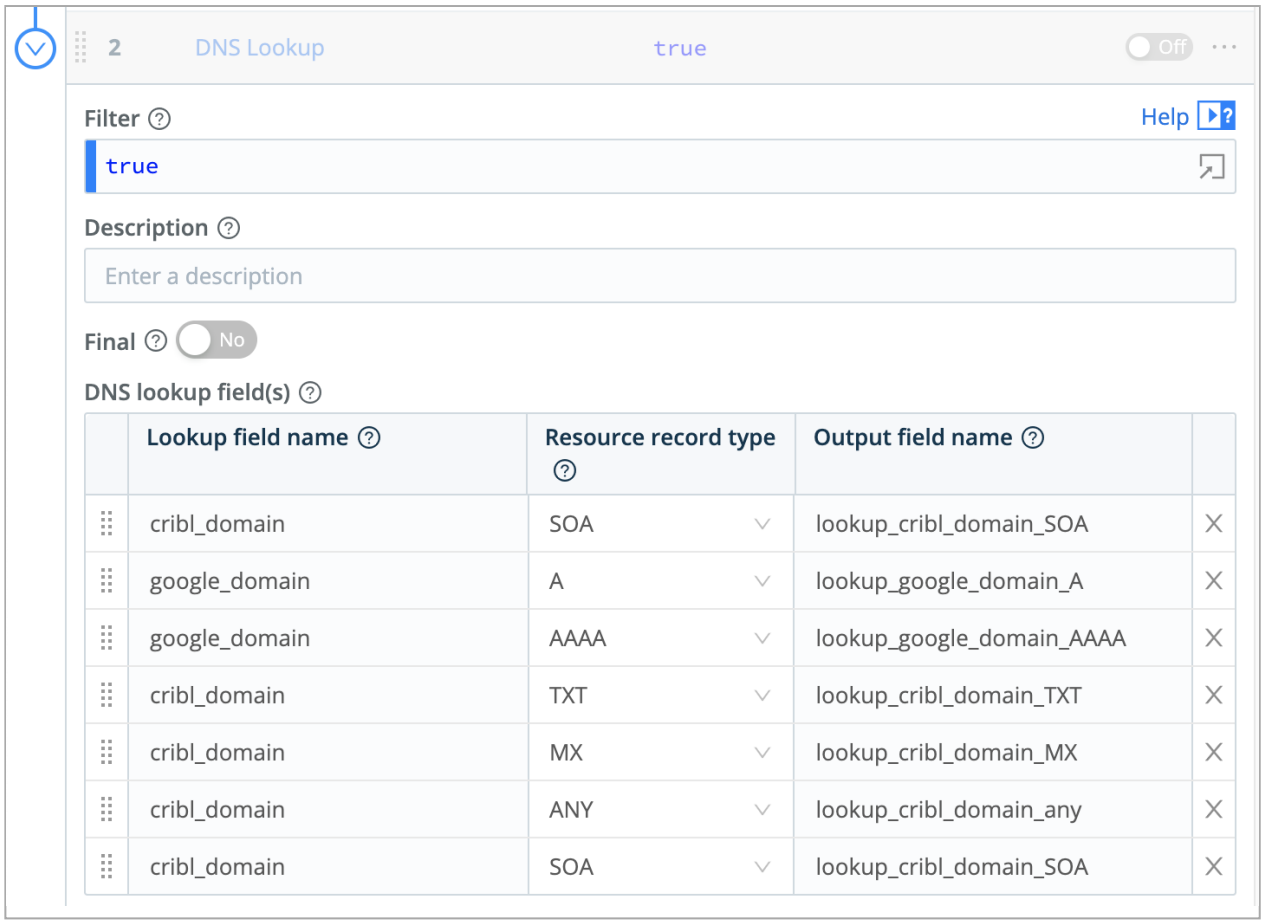
Example

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.



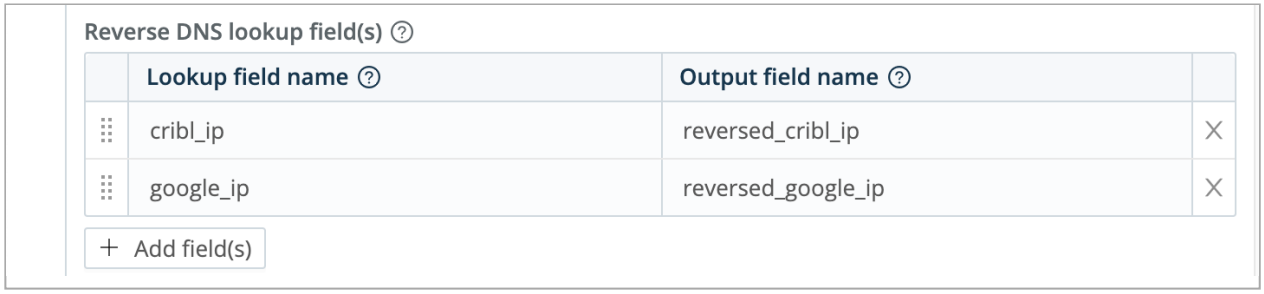
DNS Lookup: Eval Function

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.

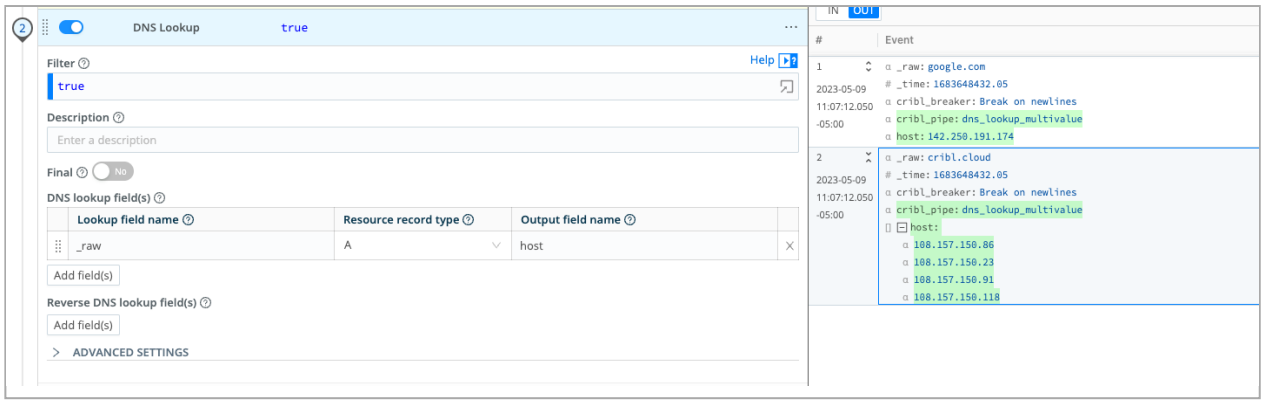


DNS Lookup: reverse lookups

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.



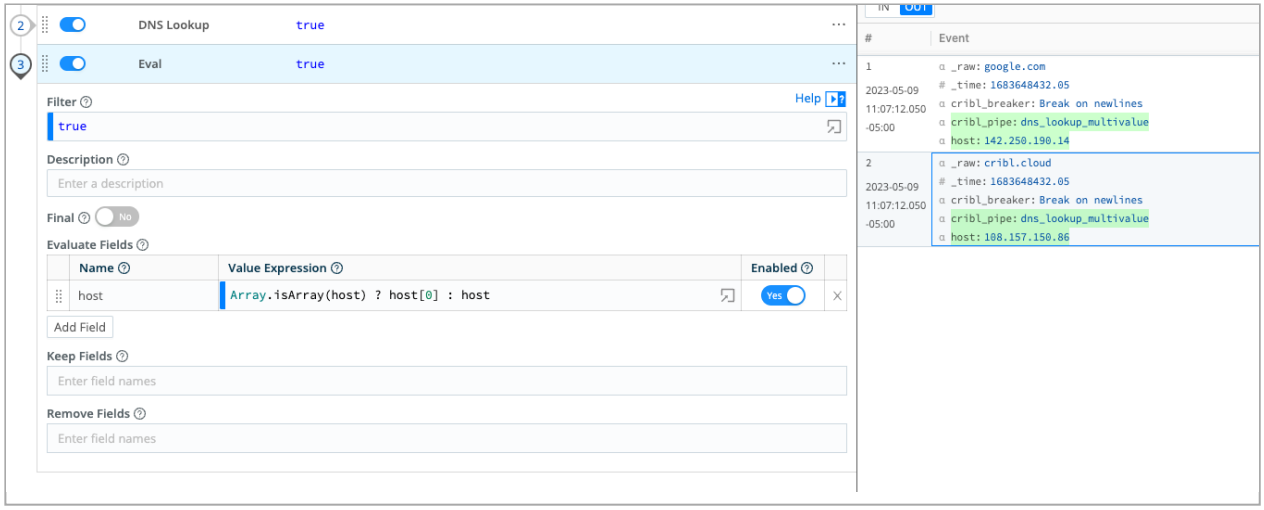
DNS lookup with multi-value result

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.

12.9. 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'`

12.10. 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

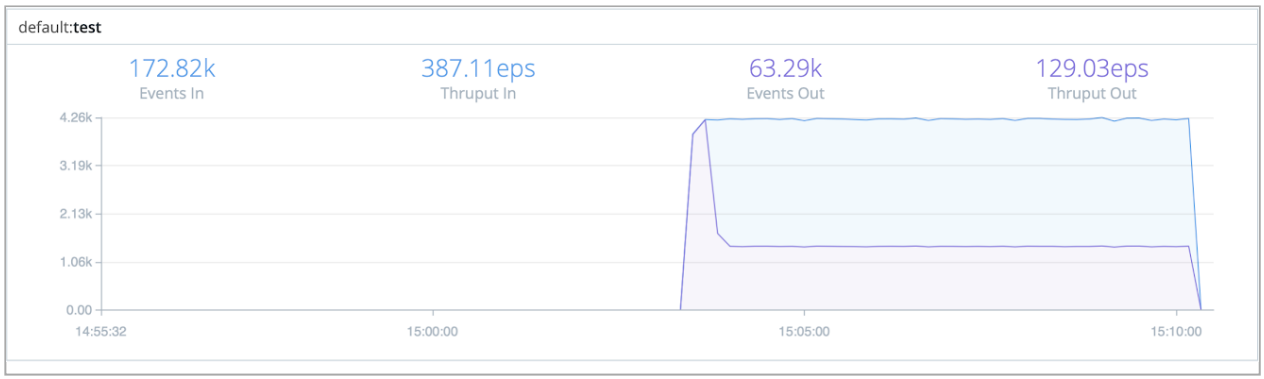
Here's an example that illustrates the effectiveness of using the Square Root sample mode.

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.

12.11. 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 Ingest-time Fields for more examples.

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.

12.12. 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.

12.13. 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.

12.14. 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`

NAME	VALUE EXPRESSION
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.

12.15. 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 links for creating and testing Grok patterns: http://grokdebug.herokuapp.com and http://grokconstructor.appspot.com/.
- Additional patterns are available here: https://github.com/logstash-plugins/logstash-patterns-core/tree/master/patterns.

12.16. 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":"Ford","models":["Fiesta","Focus","Mustang"]}}"}

Event 2:
{"_raw":"{"date":"9/25/18 9:10:13.000 PM","name":"Amrit","age":42,"cars":
{"name":"GM","models":["Trans AM","Oldsmobile","Cadillac"]}}"}

Event 3:
{"_raw":"{"date":"9/25/18 9:10:13.000 PM","name":"Amrit","age":42,"cars":
{"name":"Fiat","models":["500","Panda"]}}"}

Event 4:
{"_raw":"{"date":"9/25/18 9:10:13.000 PM","name":"Amrit","age":42,"cars":
{"name":"Blackberry","models":["KEY2","Bold Touch 9900"]}}"}
```


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**

12.17. Lookup

The Lookup Function enriches events with external fields, using lookup table files in CSV, compressed `.csv.gz`, or binary `.mmdb` 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 about 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.

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.

```
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`

```
### BEFORE:

{"_raw": "Sep 20 13:03:55 PA-VM 1,2018/09/20 13:03:58,FOOBAR,TRAFFIC,end,2049,2018/09/2
13:03:58,34.217.108.226,10.0.0.102,34.217.108.226,10.0.2.65,splunk,,,incomplete,vsys1,u
forwarding-default,2018/09/20 13:03:58,574326,1,53722,8088,53722,8088,0x400064,tcp,allo
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,
{"_raw": "Sep 20 13:03:55 PA-VM 1,2018/09/20 13:03:58,FOOBAR,THREAT,end,2049,2018/09/20
13:03:58,34.217.108.226,10.0.0.102,34.217.108.226,10.0.2.65,splunk,,,incomplete,vsys1,u
forwarding-default,2018/09/20 13:03:58,574326,1,53722,8088,53722,8088,0x400064,tcp,allo
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,


### AFTER:

{"_raw": "Sep 20 13:03:55 PA-VM 1,2018/09/20 13:03:58,FOOBAR,TRAFFIC,end,2049,2018/09/2
13:03:58,34.217.108.226,10.0.0.102,34.217.108.226,10.0.2.65,splunk,,,incomplete,vsys1,u
forwarding-default,2018/09/20 13:03:58,574326,1,53722,8088,53722,8088,0x400064,tcp,allo
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,
  "sourcetype": "pan:traffic"
  }
{"_raw": "Sep 20 13:03:55 PA-VM 1,2018/09/20 13:03:58,FOOBAR,THREAT,end,2049,2018/09/20
13:03:58,34.217.108.226,10.0.0.102,34.217.108.226,10.0.2.65,splunk,,,incomplete,vsys1,u
forwarding-default,2018/09/20 13:03:58,574326,1,53722,8088,53722,8088,0x400064,tcp,allo
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,
  "sourcetype": "pan:threat"
  }
```

Example 2: CIDR Lookups

Assign a `location` field to events if their `destination_ip` field matches a particular CIDR range.

```
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.

```
### BEFORE:

{"_raw": "Sep 20 13:03:55 PA-VM 1, 2018/09/20 13:03:58,FOOBAR,TRAFFIC,end,2049,2018/09/
13:03:58,34.217.108.226,10.0.0.102,34.217.108.226,10.0.2.65,splunk,,,incomplete,vsys1,u
forwarding-default,2018/09/20 13:03:58,574326,1,53722,8088,53722,8088,0x400064,tcp,allo
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,
  "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,2049,2018/09/
13:03:58,34.217.108.226,10.0.0.102,34.217.108.226,10.0.2.65,splunk,,,incomplete,vsys1,u
forwarding-default,2018/09/20 13:03:58,574326,1,53722,8088,53722,8088,0x400064,tcp,allo
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,
  "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,2049,2018/09/
13:03:58,34.217.108.226,10.0.0.102,34.217.108.226,10.0.2.65,splunk,,,incomplete,vsys1,u
forwarding-default,2018/09/20 13:03:58,574326,1,53722,8088,53722,8088,0x400064,tcp,allo
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,
  "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,2049,2018/09/
13:03:58,34.217.108.226,10.0.0.102,34.217.108.226,10.0.2.65,splunk,,,incomplete,vsys1,u
forwarding-default,2018/09/20 13:03:58,574326,1,53722,8088,53722,8088,0x400064,tcp,allo
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,
  "destination_ip": "10.0.0.102",
  "location": [
    "San Francisco",
    "California",
    "US",
  ]}
```

More Examples and Scenarios

More examples:

- Ingest-time Lookups.
- Lookups and Regex Magic.
- Lookups as Filters for Masks.

See also:

- Managing Large Lookups to optimize file locations for large lookup files.
- Redis Function for faster lookups using a Redis integration.

12.18. 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:

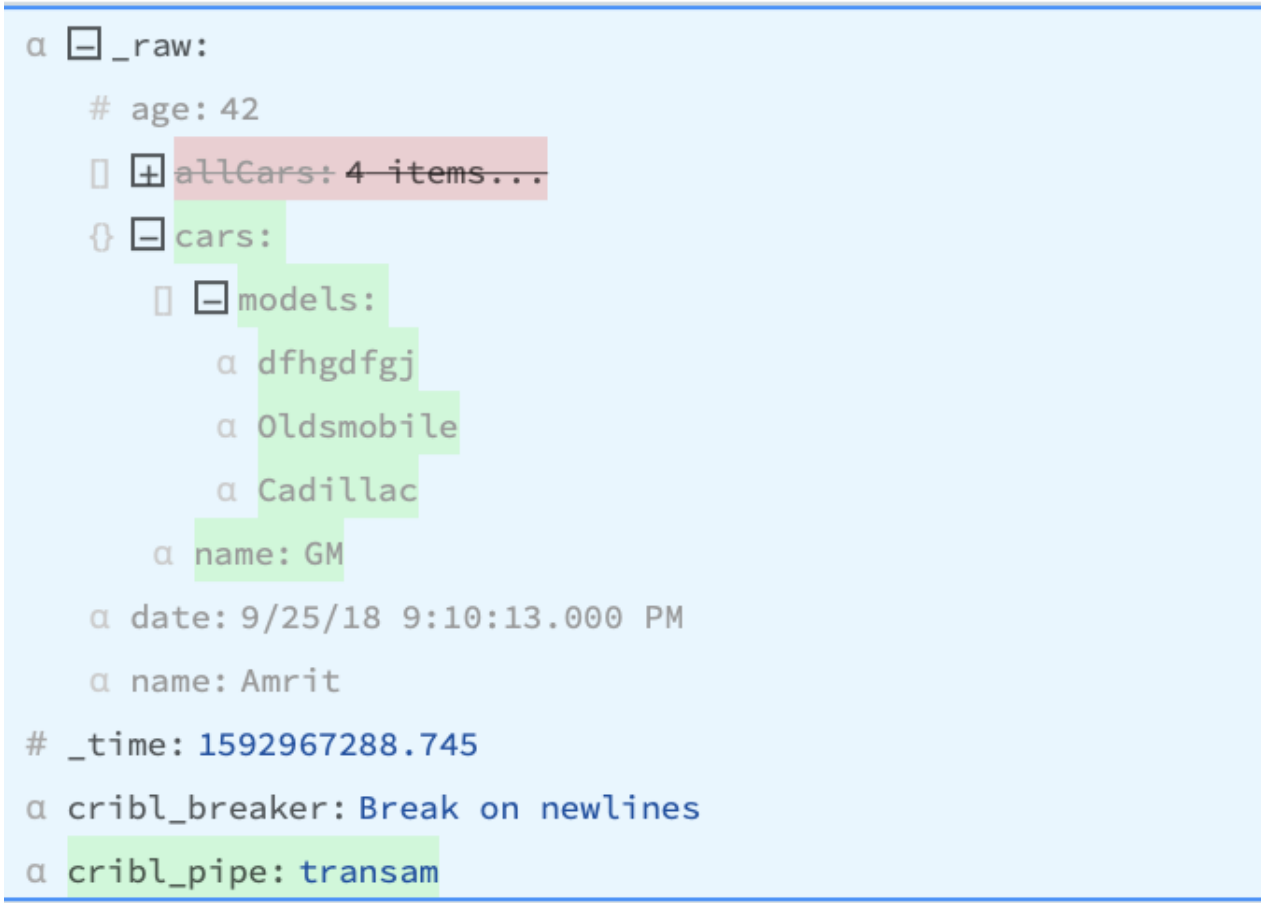
- We don't expect this to be a common case.
- Expanding the event in the replace context would have a high performance hit on the common path.
- There is a slight chance that there might be a `gN` field in the event.

> 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

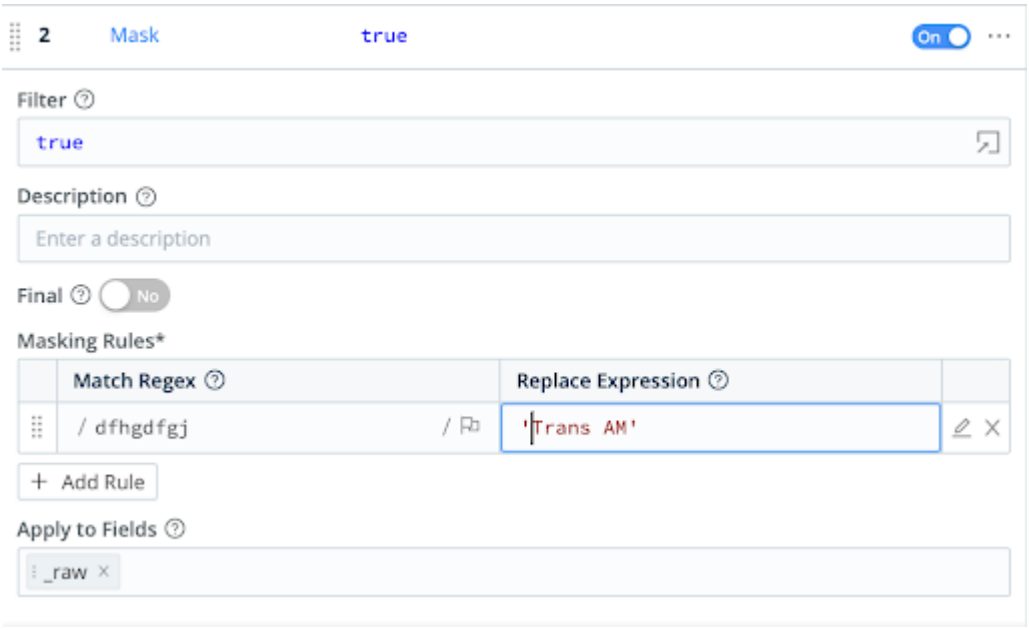
Here, we'll simply search for the string `dfhgdfgj`, and replace that value (if found) with `Trans AM`. This will help close America's muscle-car gap:



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=#########`.

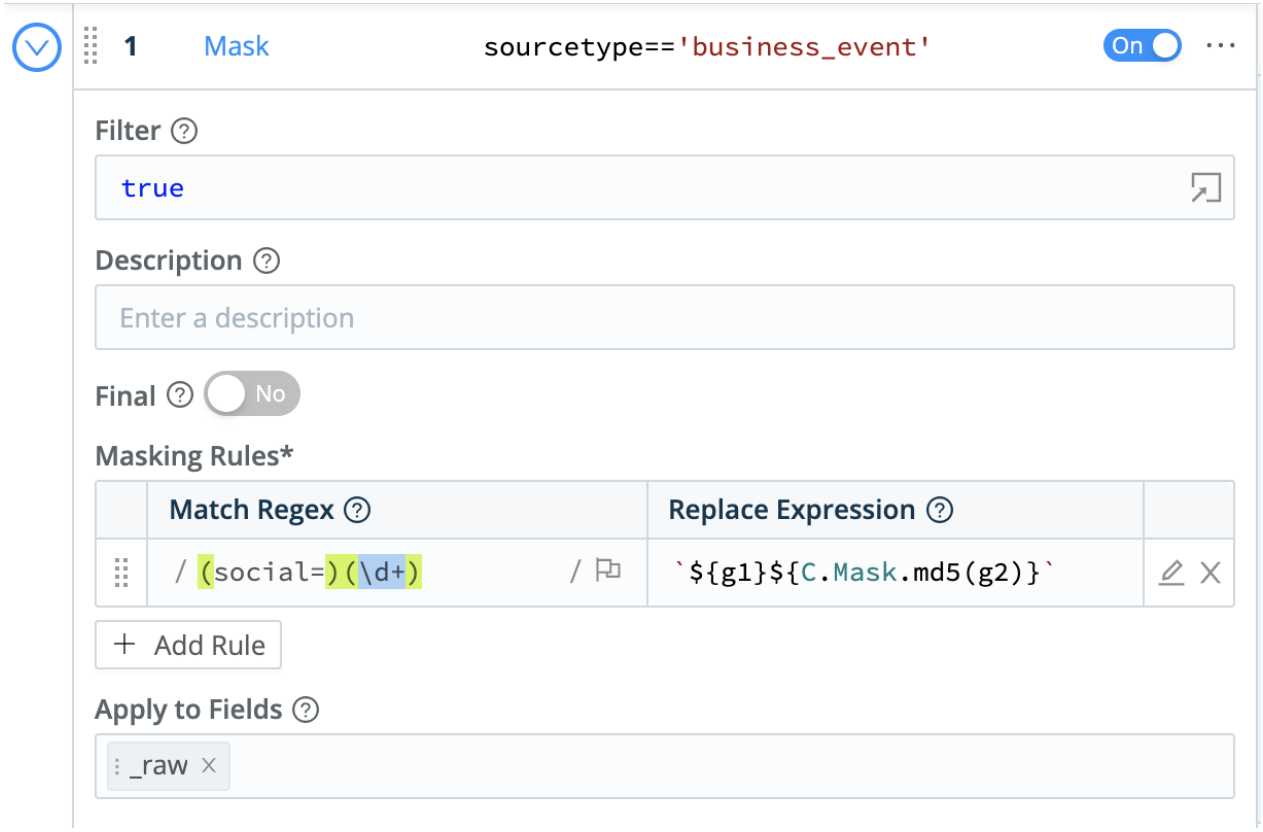
Event with unredacted SSNs

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.

Event with hashed SSNs

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.

> For further masking examples, see Masking and Obfuscation.

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
```

12.19. 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`.

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**.

> ### Double Negatives
>
> **Ignore fields** also supports negated terms. When you negate field names, the fields list is order-sensitive. E.g., `!foobar` before `foo*` means "Ignore all fields that start with `foo`, except `foobar`." However, `!foo*` before `*` means "Ignore all fields, except for those that start with `foo`."

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+\S+\s+(?<bytes>\d+)`
2. Then use Numerify.

12.20. 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.
- JSON
- K=V Pairs – selecting this type requires you to escape certain characters.
- Delimited Values – selecting this type expands the UI.
- Common Log Format (CLF)
- Extended Log File Format (ELFF)

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.

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 delimiter or quote characters. 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

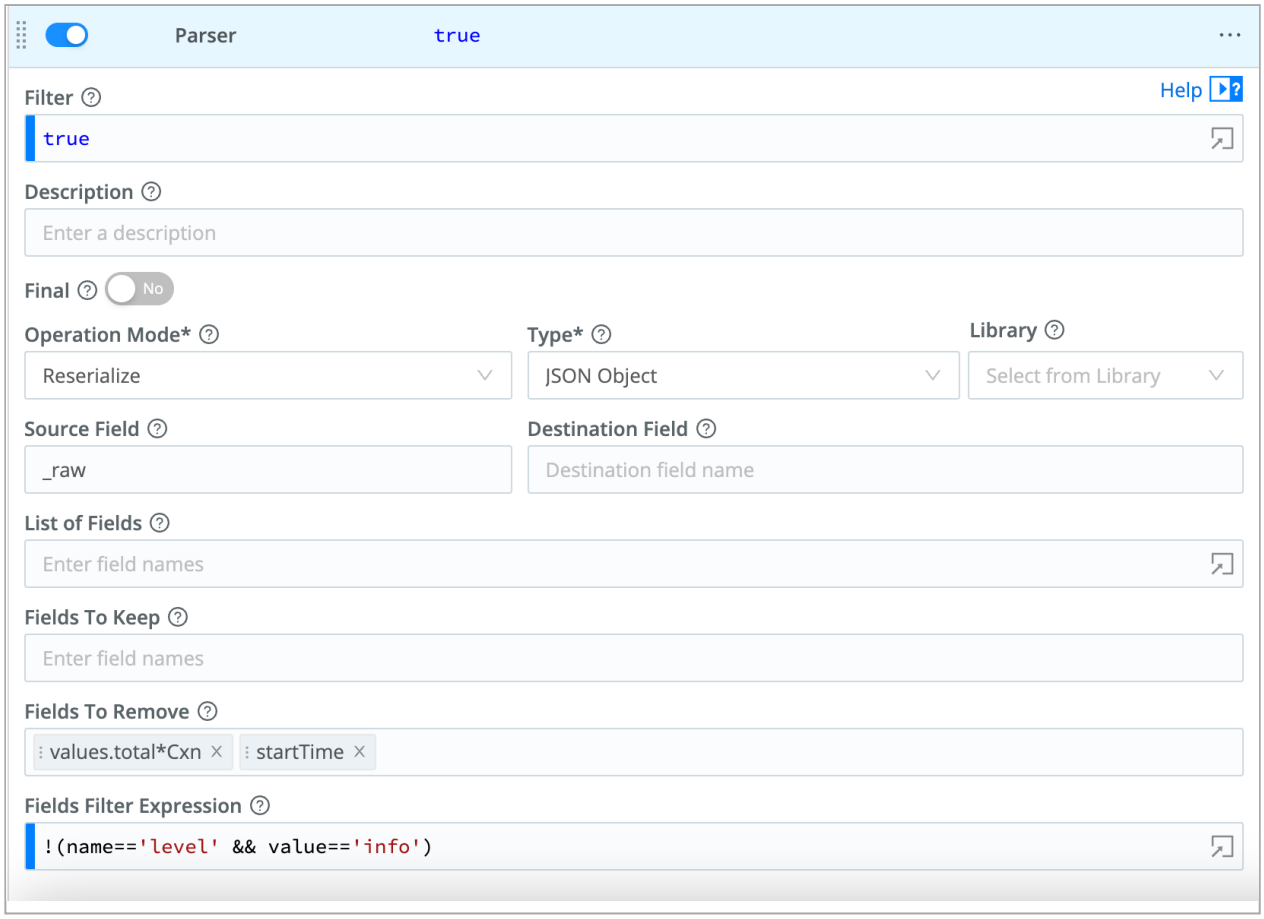
Examples

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`

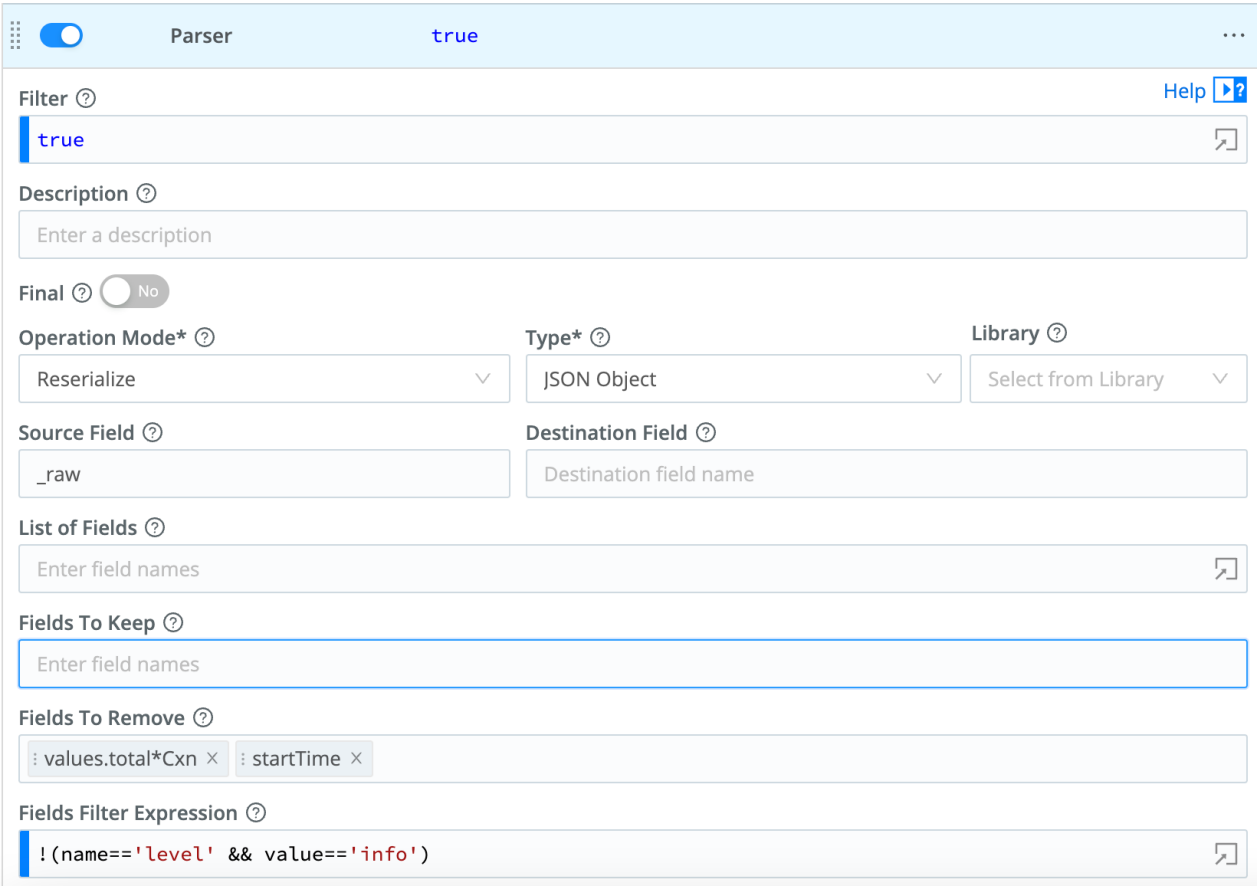
> The `index` refers to the location of a field in the array of all fields extracted by **this** Parser. It is zero-based. In the case above, `g2` and `g3` have `index` values of `7` and `8`, respectively.

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:

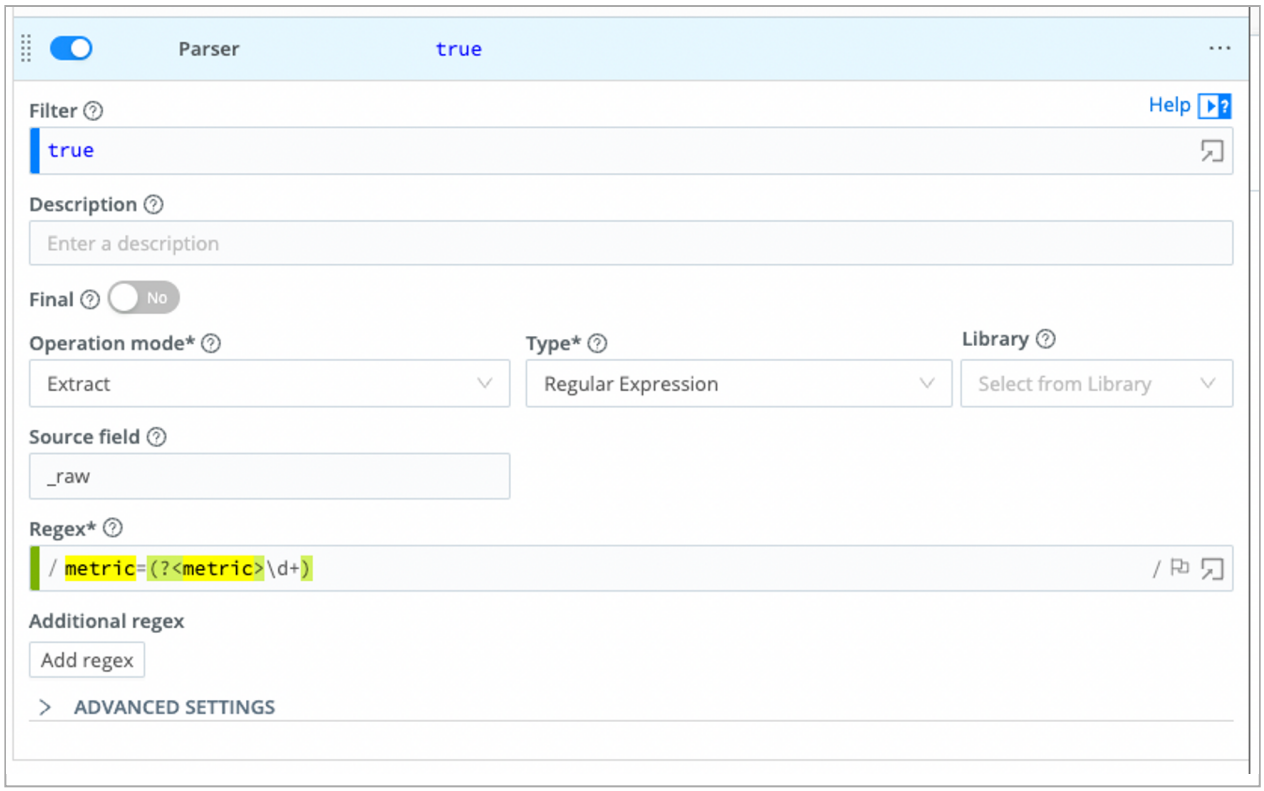
- Operation mode: `Serialize`
- Source field: [empty]
- Destination field: `bar`
- Type: `JSON`
- List of fields: [empty]
- Fields to keep: `a, b, c, d`

Example 4

Assume we have a simple event that looks like this:

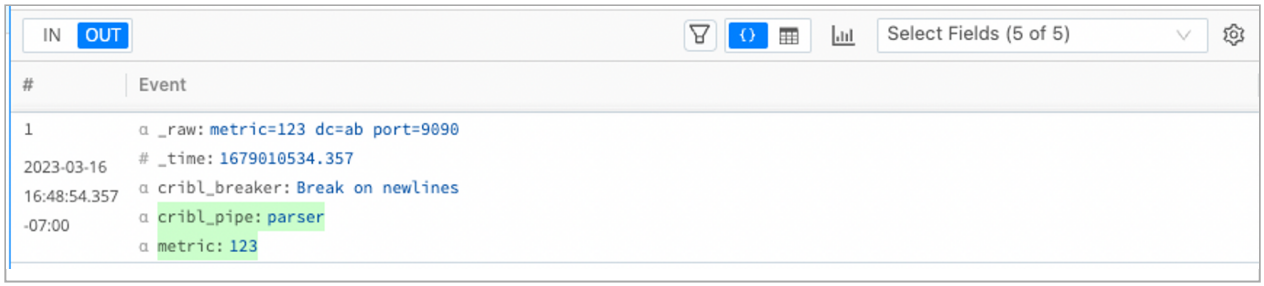
```
metric=123 dc=ab port=9090
```

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

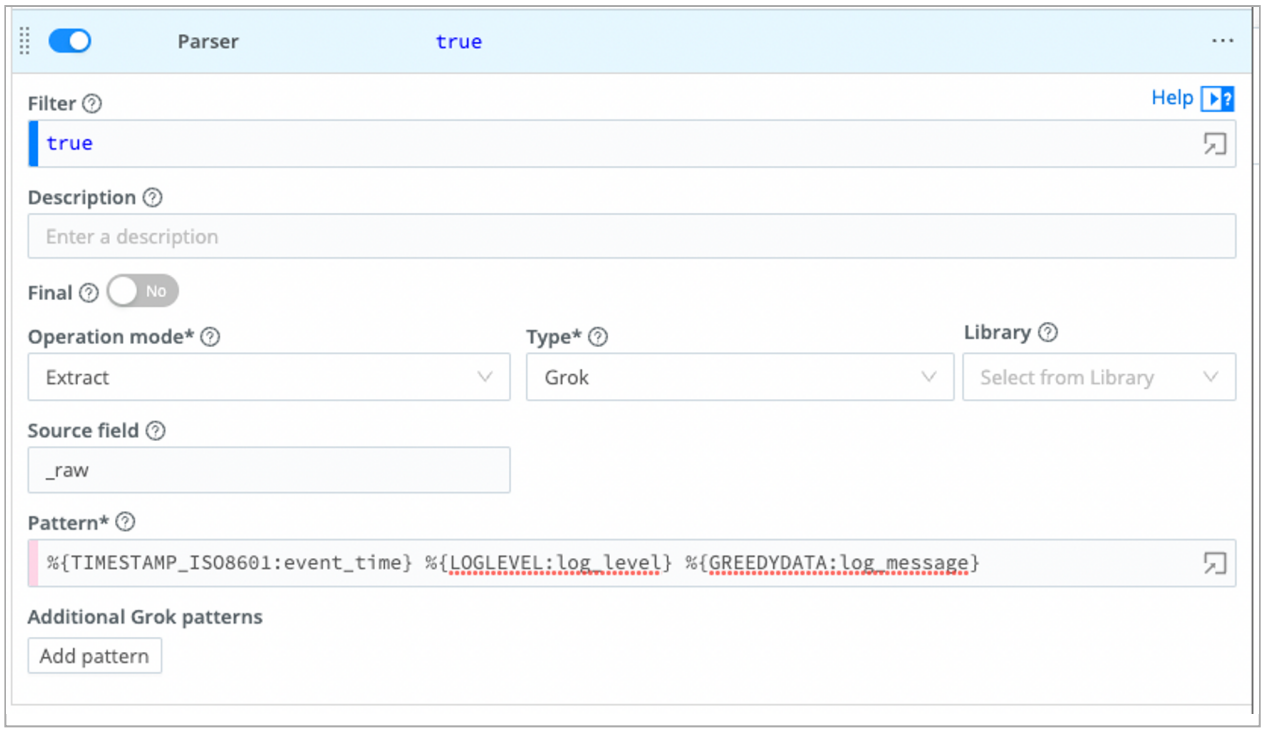
Assume we have an event that looks like this:

```
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:

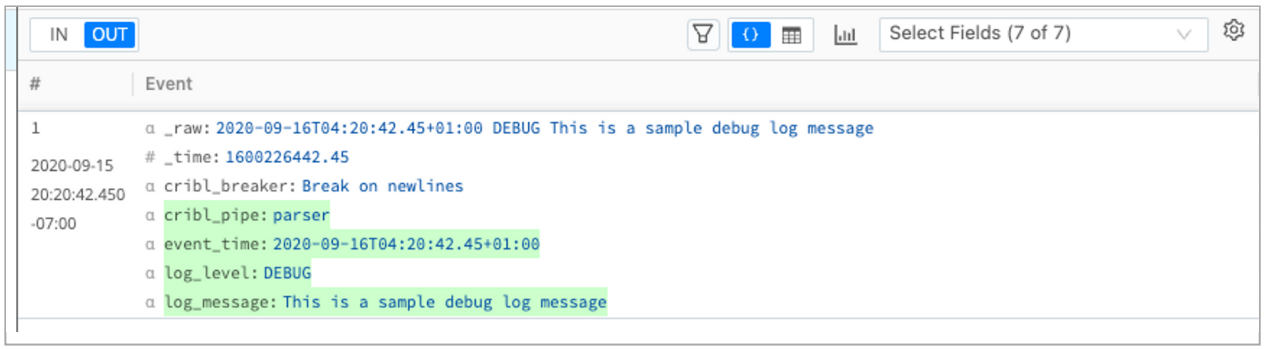
```
%{TIMESTAMP_ISO8601:event_time} %{LOGLEVEL:log_level} %{GREEDYDATA:log_message}
```

The Parser Function configuration looks like this:



Parser Function configuration for Example 5

The output will look like this:



Parser Function grok output

12.21. 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.

- **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 `!_* *`.

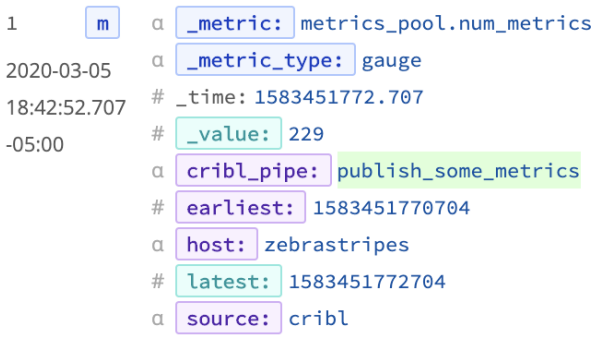
Remove Dimensions: Optional list of dimensions to associate with every extracted metric value. Leave blank if this function is used to process output from the Aggregation function as dimensions will be automatically discovered. If this Function is used to process output from the Aggregations Function, leave this field blank, because 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:ethernet1/2,destination_port:443
metric.10.10.12.192.bytes_rcvd:0|c|#destination_ip:160.177.222.249,inbound_interface:ethernet1/2,destination_port:443
metric.10.10.12.192.pkts_sent:1|c|#destination_ip:160.177.222.249,inbound_interface:ethernet1/2,destination_port:443
metric.10.10.12.192.pkts_rcvd:0|c|#destination_ip:160.177.222.249,inbound_interface:ethernet1/2,destination_port:443
```

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`.

12.22. 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 (e.g., 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 production 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.

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, e.g.: `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.

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 (e.g., 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 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.

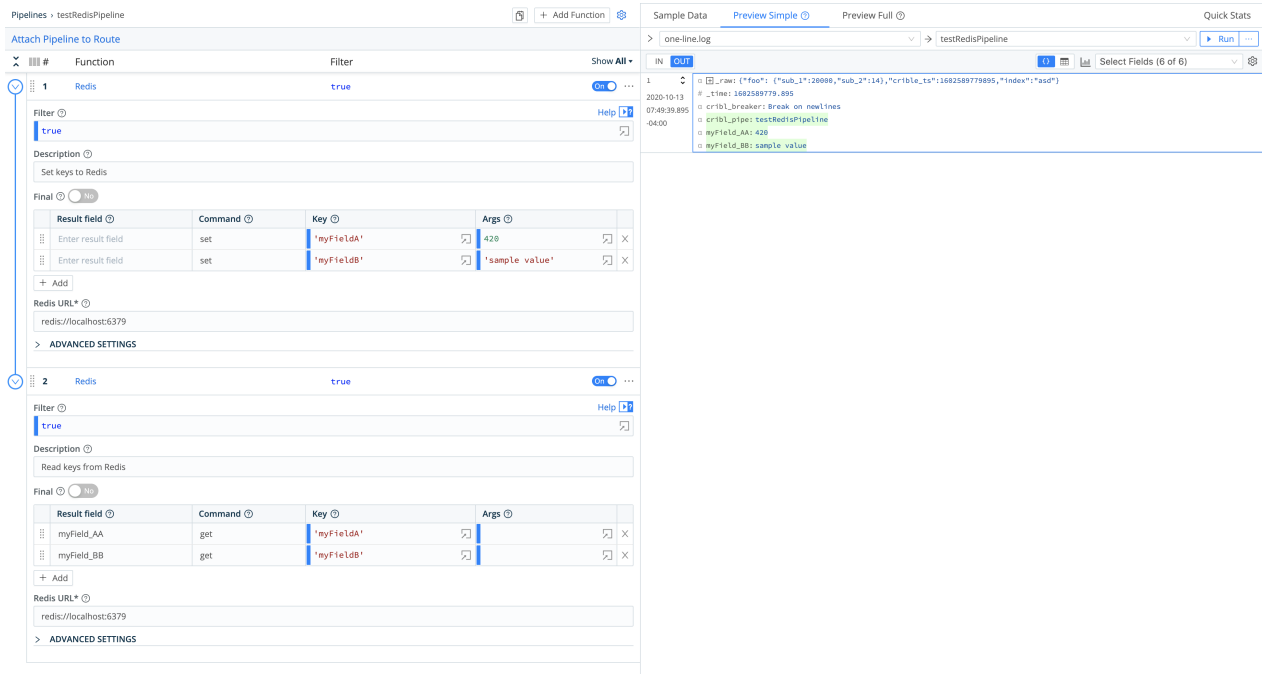
You should select whichever of these mechanisms best suits your environment. Does it make more sense to impose the heavier burden on the Redis server (as in **Default** mode) or on Cribl Edge (as in **Broadcast** mode)?

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

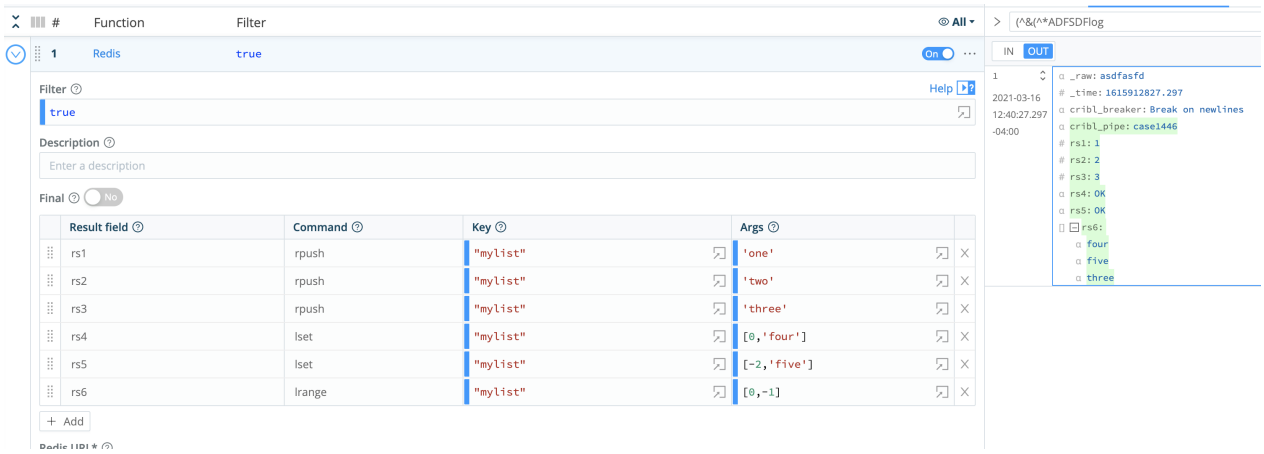
Description: `Read keys from Redis`

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"
      }
    ]
  }
}
```

12.23. Regex Extract

The Regex Extract Function extracts fields using regex named 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 not** be added to events, and **will not** exit the Pipeline.

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, e.g.: `(?<foo>bar)`. Can contain special `_NAME_N` and `_VALUE_N` capturing groups, which extract **both the name and value** of a field, e.g.: `(?<_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`.

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. E.g., 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>`.

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. (E.g., 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"]`.)

Examples

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: Key-Value Pairs from Multiple Fields

Use this sample:

```
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=172.20.115.42
sec_intel_event=No mac_address=00:00:00:00:00:00 dest_bytes=3746
dest_autonomous_system=0 security_context=00000000000000000000000000000000
src_port=41925 web_app=Unknown url=https://outlook.ssg.petsmart.com
url_reputation="Risk unknown" first_pkt_sec=1543598207 vlan_id=0 ssl_flow_error=0
ssl_actual_action=Unknown has_ipv6=1 monitor_rule_6=N/A monitor_rule_7=N/A
monitor_rule_4=N/A monitor_rule_5=N/A monitor_rule_2=N/A monitor_rule_3=N/A
ips_count=0 monitor_rule_1=N/A dest_tos=0 src_ip=192.168.228.5 referenced_host=""
iface_ingress=DMZ3.30 monitor_rule_8=0 event_subtype=1 fw_rule_reason=N/A
event_type=1003 ssl_version=Unknown dns_resp_id=0 sensor=ssg-inet-fpr-ftd-fw01
sec_zone_egress=Inside src_tos=0 client_app="SSL client" snmp_in=0 user=Unknown
ssl_flow_messages=0 iface_egress=inside http_referrer="" src_pkts=0 event_desc="Flow
Statistics" event_usec=0 client_version="" fw_rule_action=Allow
ssl_cert_fingerprint=0000000000000000000000000000000000000000 ssl_url_category=0
file_count=0 sec_zone_ingress=DMZ3 instance_id=6 src_bytes=1013
src_ip_country=unknown ssl_cipher_suite=TLS_NULL_WITH_NULL_NULL user_agent=""
http_response=0 src_mask=0 dest_mask=0 sec_intel_ip=N/A netbios_domain=""
tcp_flags=0 dns_rec_id=0 fw_policy="SSG INET Access Control Policy"
last_pkt_sec=1543598207 legacy_ip_address=0.0.0.0 ip_proto=TCP connection_id=21378
dest_pkts=0 app_proto=HTTPS ssl_flow_status=Unknown ssl_rule_id=0
ssl_session_id=0000000000000000000000000000000000000000000000000000000000000000
dns_query="" rec_type_desc="Connection Statistics" url_category=Unknown
fw_rule="Outbound Web" src_autonomous_system=0 ssl_flow_flags=0 ip_layer=0
event_sec=1543598205 ssl_ticket_id=00000000000000000000000000000000000000000000
sinkhole_uuid=00000000-0000-0000-0000-000000000000 dest_ip_country=unknown
ssl_expected_action=Unknown num_ioc=0 dns_ttl=0
ssl_policy_id=00000000000000000000000000000000 ssl_server_name=""
```

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 2 results

Example 3: Multi-Stage Extraction, Complex Events

This example builds on the syntax in Example 2, to tackle a more complex event structure.

In the right **Sample Data** pane, click **Paste** and insert the following sample:

```
<134>1 2020-12-22T17:06:08Z CORP_INT_NLB CheckPoint 18160 - [action:"Accept";
conn_direction:"Internal"; flags:"4606212"; ifdir:"inbound"; ifname:"bond2.1025";
logid:"0"; loguid:"{0x5fe25889,0x0,0x80ad57cd,0xeb91c0c3}"; origin:"192.168.20.54";
originsicname:"CN=TST32-VSX0-FW-DC-01_tst302-shd,O=CORP-SEC-SHRD-CMA..t7xpcz";
sequencenum:"3"; time:"1608656768"; version:"5"; __policy_id_tag:"product=VPN-1 &
FireWall-1[db_tag={15E4B45A-663B-5B49-BD59-
CD9B9F21AA16};mgmt=SHRDFW01CON;date=1608236862;policy_name=TEST-SHRD-POL\]";
dst:"192.168.79.20"; log_delay:"1608656768"; layer_name:”TEST-SHRD-POL Security";
layer_uuid:"e914c2f3-d7bd-4a77-8e7a-7a5e403447aa"; match_id:"1"; parent_rule:"0";
rule_action:"Accept"; rule_uid:"001ab86d-d201-4b61-9b64-0fede1a9f059"; product:"VPN-
1 & FireWall-1"; proto:"17"; s_port:"45519"; service:"123"; service_id:"ntp-udp";
src:"192.168.79.22"; ]
```

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:


```
α _raw: <134>1 2020-12-22T17:06:08Z CORP_INT_NLB CheckPoint 18160 - [action:"Accept"; conn_directi
        on:"Internal"; flags:"4606212"; ifdir:"inbound"; ifname:"bond2.1025"; logid:"0"; logui
        d:"{0x5fe25889,0x0,0x80ad57cd,0xeb91c0c3}"; origin:"192.168.20.54"; originsicname:"CN=TST3
        2-VSX0-FW-DC-01_tst302-shd,O=CORP-SEC-SHRD-CMA..t7xpcz"; sequencenum:"3"; time:"160865676
        8"; version:"5"; __policy_id_tag:"product=VPN-1 & FireWall-1[db_tag={15E4B45A-663B-5B49-BD
        59-CD9B9F21AA16};mgmt=SHRDFW01CON;date=1608236862;policy_name=TEST-SHRD-POL\]"; dst:"192.1
        68.79.20"; log_delay:"1608656768"; layer_name:"TEST-SHRD-POL Security"; layer_uuid:"e914c2
        f3-d7bd-4a77-8e7a-7a5e403447aa"; match_id:"1"; parent_rule:"0"; rule_action:"Accept"; rule
        _uid:"001ab86d-d201-4b61-9b64-0fede1a9f059"; product:"VPN-1 & FireWall-1"; proto:"17"; s_p
        ort:"45519"; service:"123"; service_id:"ntp-udp"; src:"192.168.79.22"; ] Show less
# _time: 1608656768
α action: "Accept"
α conn_direction: "Internal"
α cribl_breaker: Break on newlines
α cribl_pipe: asfasfdasfd
α dst: "192.168.79.20"
α flags: "4606212"
α ifdir: "inbound"
α ifname: "bond2.1025"
α layer_name: "TEST-SHRD-POL Security"
α layer_uuid: "e914c2f3-d7bd-4a77-8e7a-7a5e403447aa"
α log_delay: "1608656768"
α logid: "0"
α loguid: "{0x5fe25889,0x0,0x80ad57cd,0xeb91c0c3}"
α match_id: "1"
α origin: "192.168.20.54"
α originsicname: "CN=TST32-VSX0-FW-DC-01_tst302-shd,O=CORP-SEC-SHRD-CMA..t7xpcz"
α parent_rule: "0"
α policy_id_tag: "product=VPN-1 & FireWall-1[db_tag={15E4B45A-663B-5B49-BD59-CD9B9F21AA16}
α product: "VPN-1 & FireWall-1"
α proto: "17"
α rule_action: "Accept"
α rule_uid: "001ab86d-d201-4b61-9b64-0fede1a9f059"
```

Example 3 results

For further examples, see Using Cribl to Analyze DNS Logs in Real Time – Part 2.

12.24. 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

See Regex Filtering for examples.

12.25. 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')`

12.26. 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`.

12.27. 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:

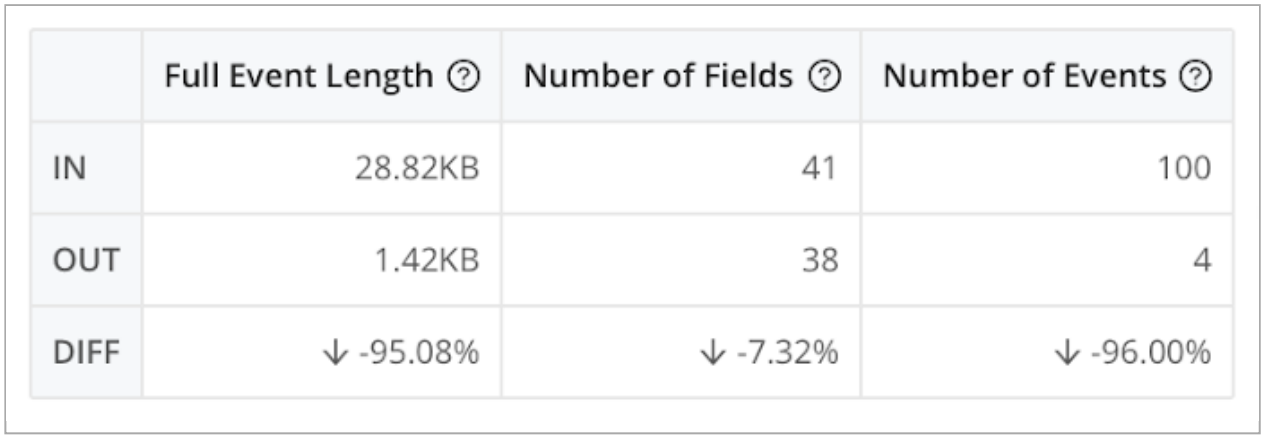
- **Filter**: Filter expression matching events to be sampled. Use `true` to match all.
- **Sampling rate**: Enter an integer `N`. (Defaults to `1`.) Sampling will pick `1`/`N` events matching this rule.

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

See Sampling for examples.

12.28. 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: Required for `CSV`, `ELFF`, `CLF`, and `Delimited values` Types. (All other formats support wildcard field lists.)

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`

12.29. 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 `300`.

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.

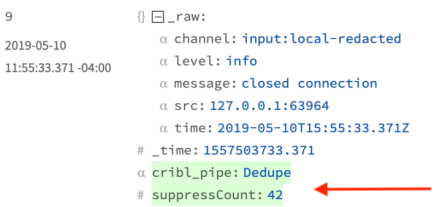
Suppression period timeout: The number of suppression periods of inactivity before a cache entry is considered idle. This defines a multiple of the **Suppression period (sec)** value. Before changing the default `2`, contact Cribl Support to understand the implications.

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.

12.30. 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.

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
eceived":"0"}
```

> 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.

12.31. Trim Timestamp

The Trim Timestamp Function removes timestamp patterns from events, and (optionally) stores them in a specified field.

This Function looks for a timestamp pattern that exists between the characters indicated by numeric `timestartpos` and `timeendpos` fields. It removes `timestartpos` and `timeendpos` along with the timestamp pattern.

> The Trim Timestamp Function, in current Cribl Edge versions, removes timestamps only from events whose `timestartpos` value is set to `0`. If you need to strip timestamps from arbitrary postions within events, instead use an 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 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

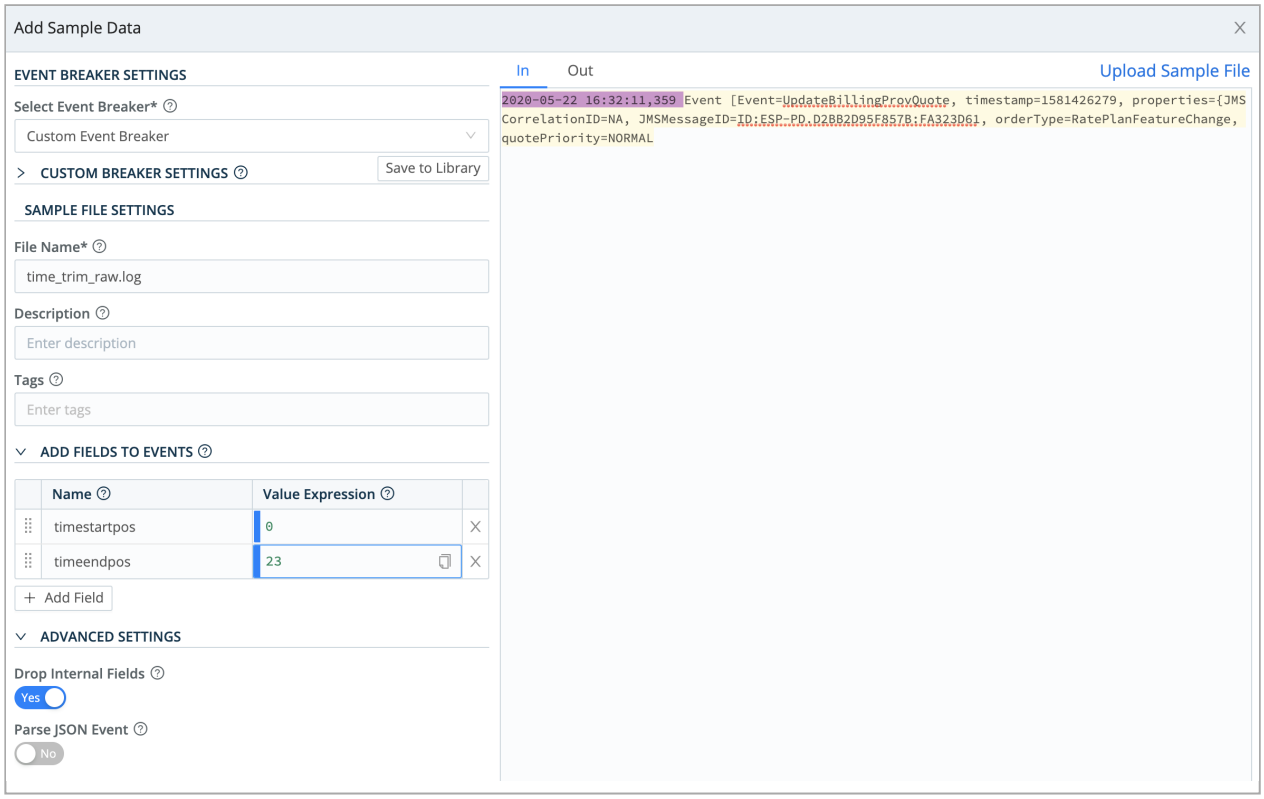
Remove the timestamp pattern (indicated by `timestartpos` and `timeendpos`) from `_raw`, and stash it in a field called `time_field`.

Field name: `time_field`

Example event before:

```
{"_raw": "2020-05-22 16:32:11,359 Event [Event=UpdateBillingProvQuote,
timestamp=1581426279, properties={JMSCorrelationID=NA, JMSMessageID=ID:ESP-
PD.D2BB2D95F857B:FA323D61, orderType=RatePlanFeatureChange, quotePriority=NORMAL}",
"timestartpos":0,
"timeendpos":23
}
```

To create this example payload, we selected **Sample Data** > **Paste**, pasted the `_raw` field's original contents into the resulting modal, and then added the two required position fields:

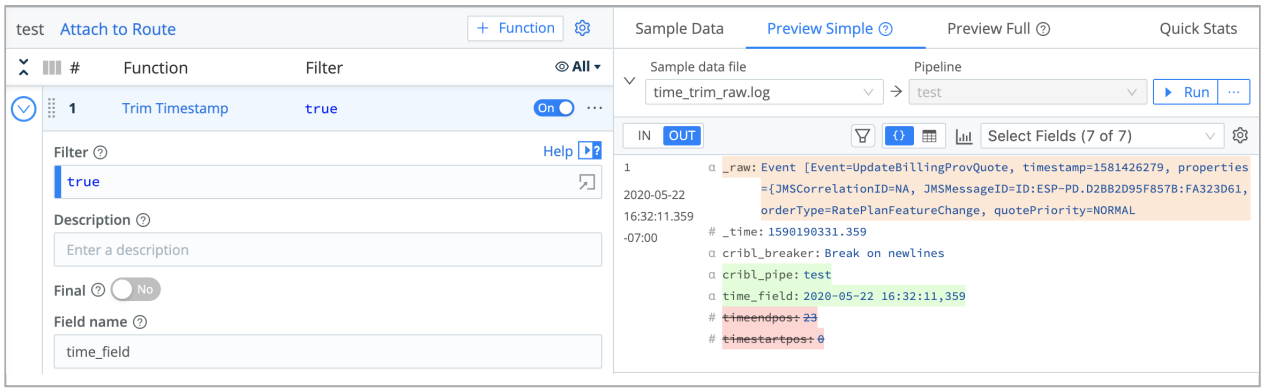


Example event setup

Example event after:

```
{"_raw": "Event [Event=UpdateBillingProvQuote, timestamp=1581426279, properties=
{JMSCorrelationID=NA, JMSMessageID=ID:ESP-PD.D2BB2D95F857B:FA323D61,
orderType=RatePlanFeatureChange, quotePriority=NORMAL}",
"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

12.32. 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:

```
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 --system --deserialize 28
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`

```
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 --system --deserialize 28

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]
```

12.33. 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>
```

12.34. 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="ethernet1/2",destination_port="443
60

# TYPE perf_192_168_1_248_bytes_rcvd counter
metric_192_168_1_248_bytes_rcvd
{destination_ip="160.177.222.249",inbound_interface="ethernet1/2",destination_port="443
0

# TYPE perf_192_168_1_248_pkts_sent counter
metric_192_168_1_248_pkts_sent
{destination_ip="160.177.222.249",inbound_interface="ethernet1/2",destination_port="443
1

# TYPE perf_192_168_1_248_pkts_rcvd counter
metric_192_168_1_248_pkts_rcvd
{destination_ip="160.177.222.249",inbound_interface="ethernet1/2",destination_port="443
0
```

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.

12.35. 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. Knowledge Objects

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:

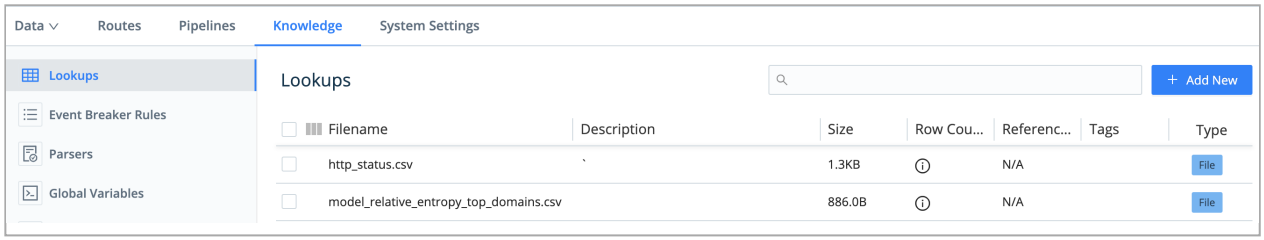
- Lookups
- Event Breakers
- Parsers
- Global Variables
- Regexes
- Grok Patterns
- Schemas (JSON)
- Parquet Schemas
- Database Connections (currently supported only in Cribl Stream)
- AppScope Configs

13.1. 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. You can access the Lookups library, which provides a management interface for all lookups, from Cribl Edge's top nav under **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

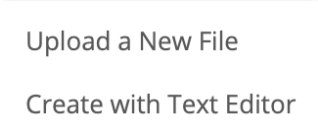
How Does the Library Work

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.

Adding Lookup Files

To upload or create a new lookup file, click **New Lookup File**, then click the appropriate option from the drop-down.

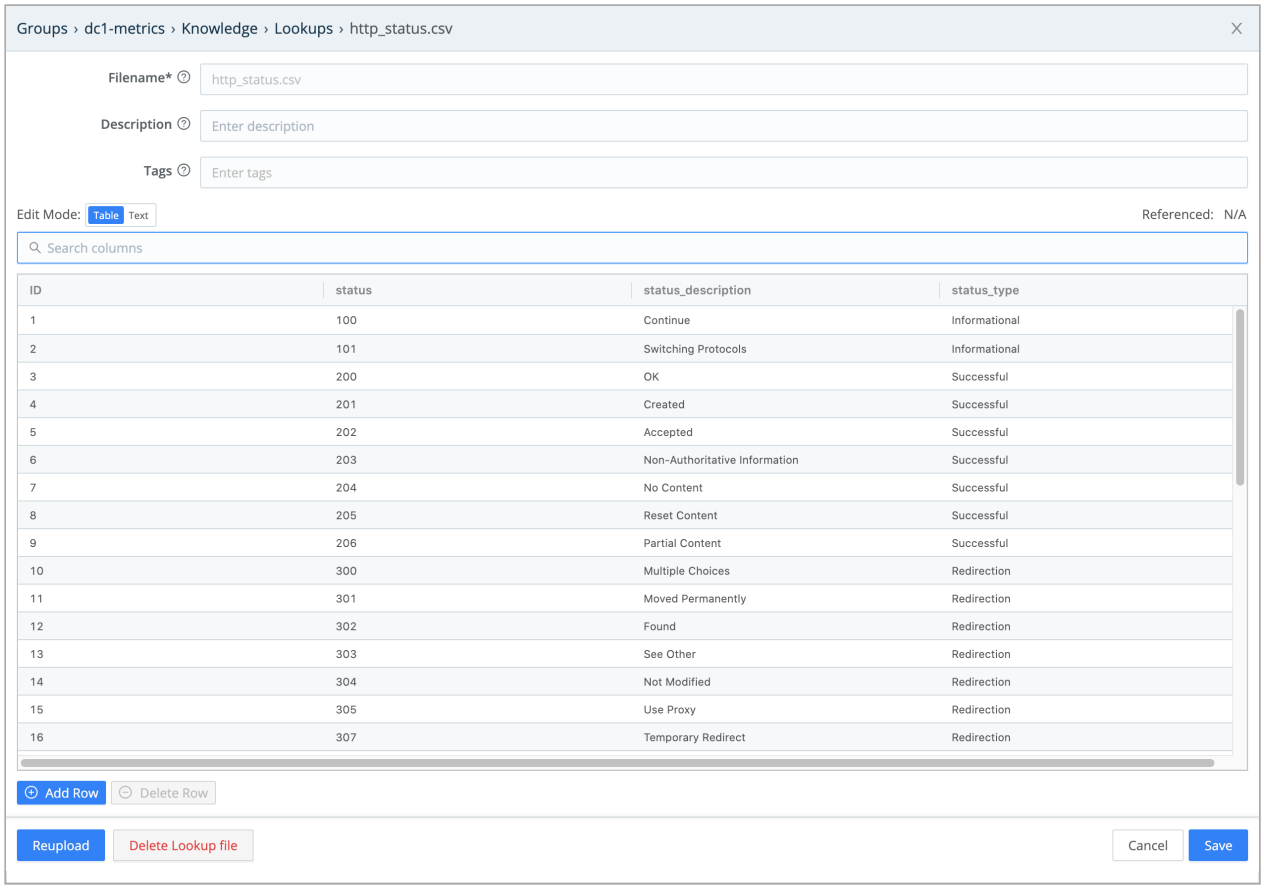


Adding a lookup file

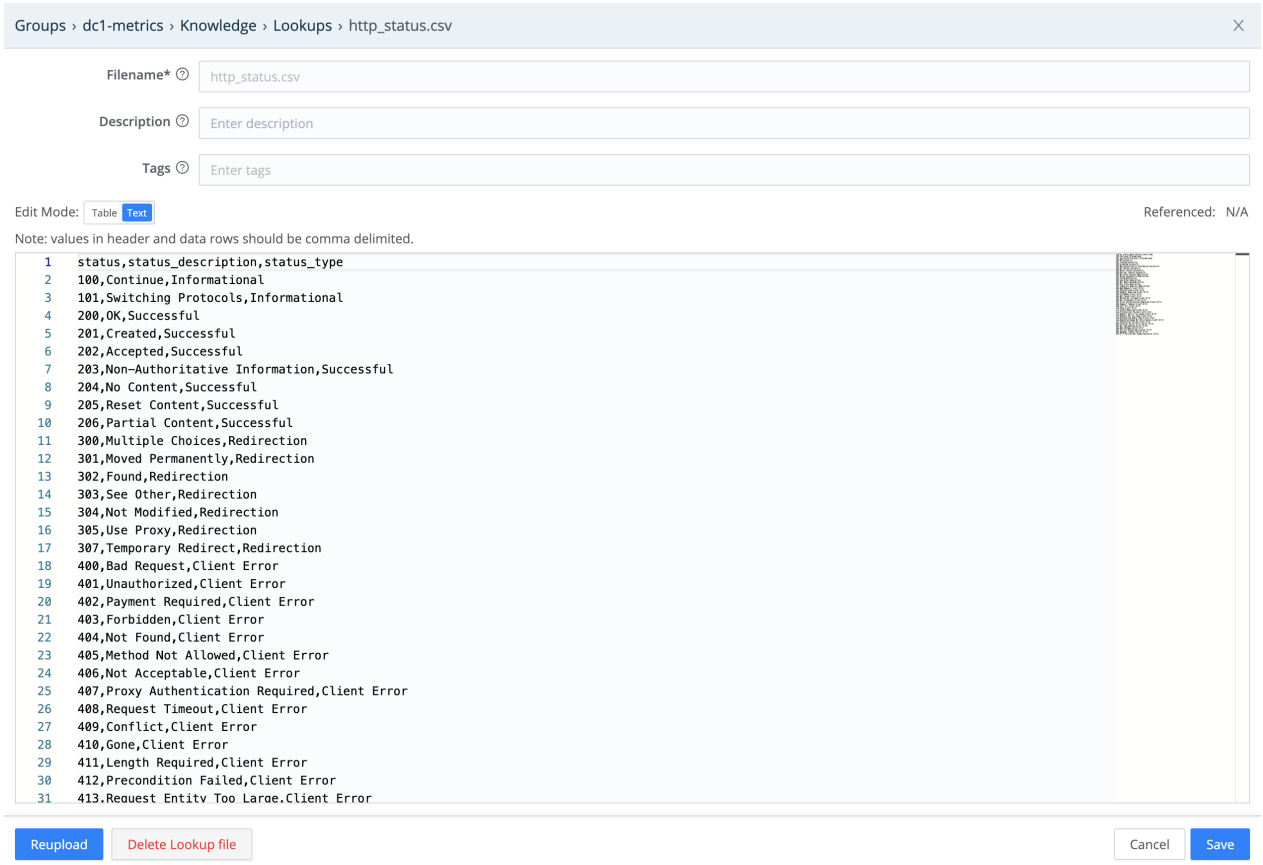
Modifying Lookup Files

To re-upload, expand, edit, or delete an existing `.csv` or `.gz` lookup file, click 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

> **Editing 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:

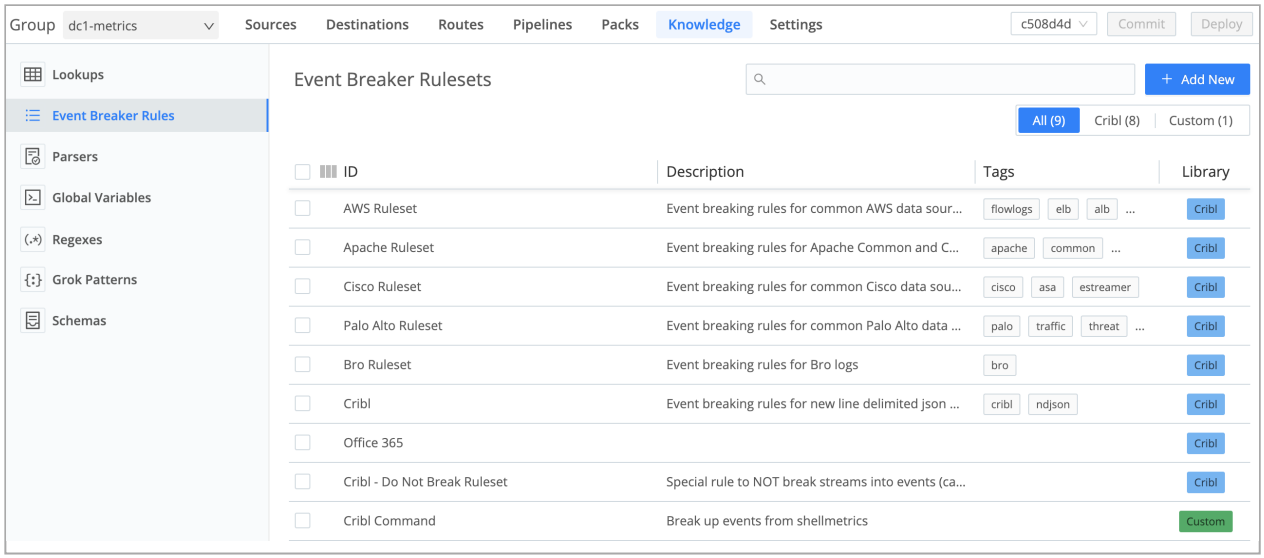
- Lookup Function.
- Ingest-time Lookups use case.
- Managing Large Lookups use case.
- Redis Function for faster lookups using a Redis integration (bypasses the Lookups Library).

13.2. 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.

Accessing Event Breakers

You access the Event Breakers management interface from Cribl Edge's top nav under **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

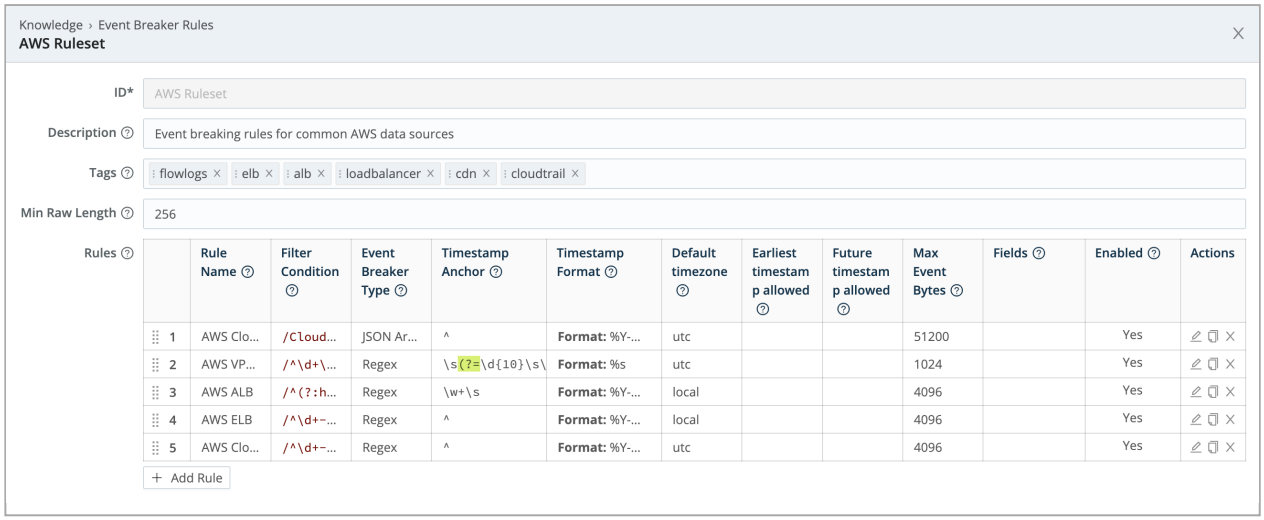
Limitations

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

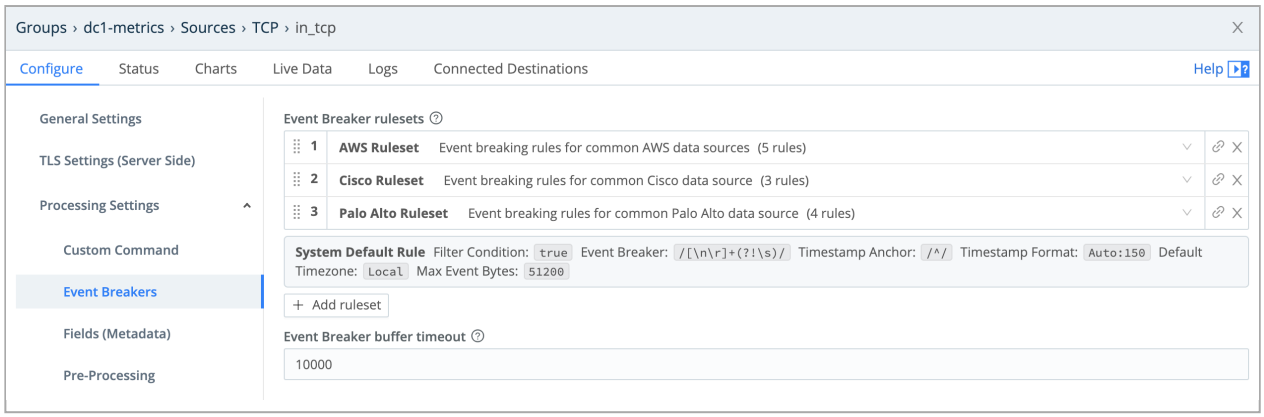
Rules within a ruleset are ordered and evaluated top->down. One or more rulesets can be associated with a Source, and these rulesets are also evaluated top->down. For a stream from a given Source, the first matching rule goes into effect.

```
Ruleset A
  Rule 1
  Rule 2
  ...
  Rule n

...

Ruleset B
  Rule Foo
  Rule Bar
  ...
  Rule FooBar
```

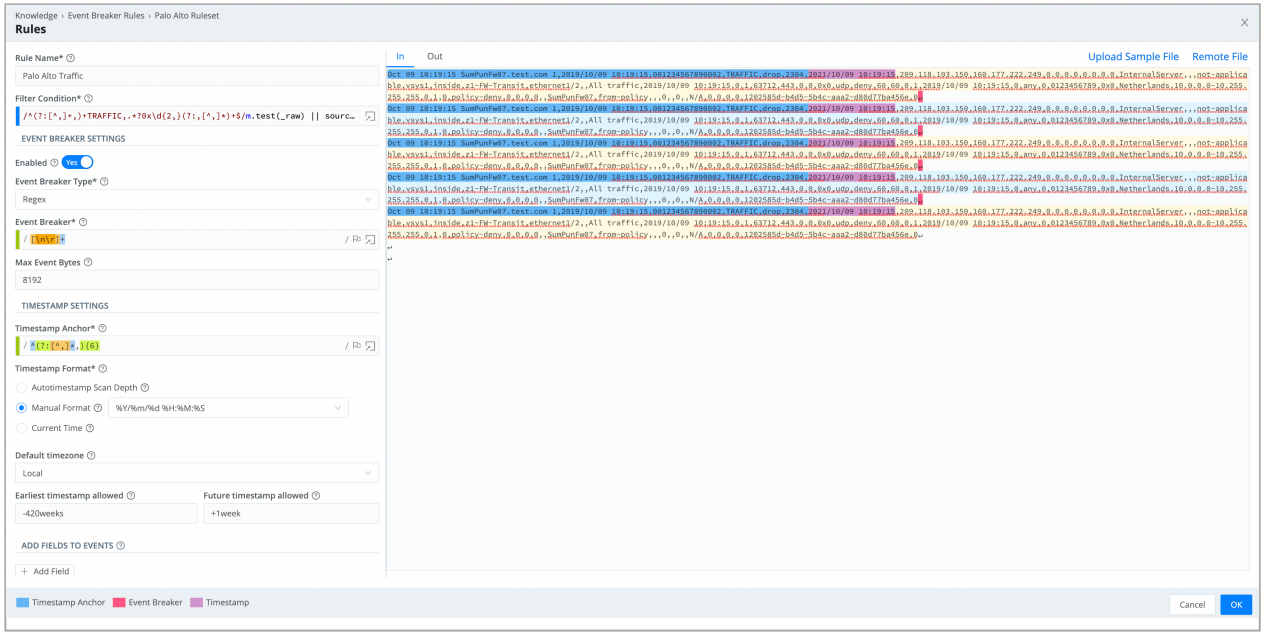
An example of multiple rulesets associated with a Source:



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

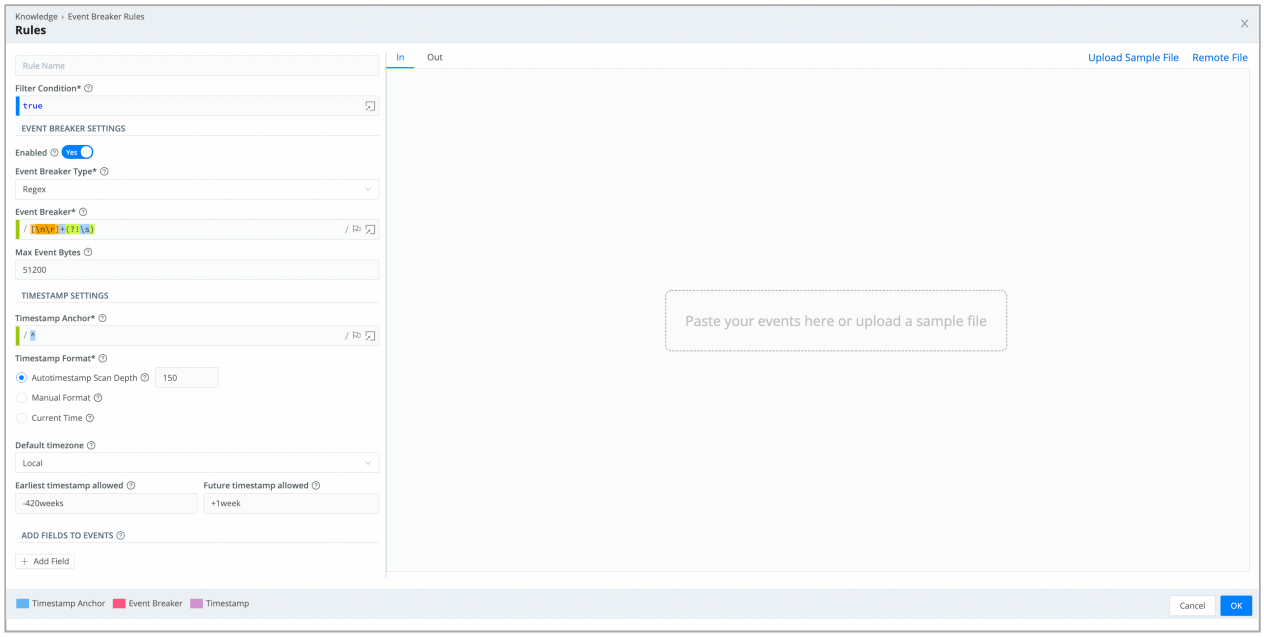
The system default rule functionally sits at the bottom of the ruleset/rule hierarchy (but is built-in and not displayed on the Event Breakers page), and goes into effect if there are no matching rules:

- 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), click **Add Ruleset** to create a new Event Breaker ruleset. Click **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

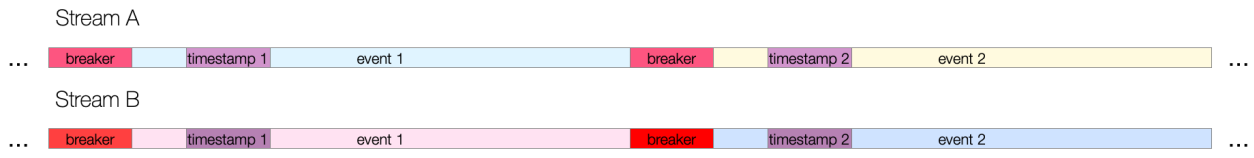
After a breaker pattern has been selected, it will apply on the stream **continuously**. See below for specific information on different Event Breaker Types.

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.)

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 **Max Event Bytes** has been reached.

The highest **Max Event Bytes** value that you can set is about 128 MB (`134217728` bytes). Events exceeding the maximum will be split into separate events, but left unbroken. Cribl Edge will set these events' `__isBroken` internal field to `false`.

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+)`

```
--- input ---
2020-05-19 16:32:12 moen3628 ipsum[5213]: Use the mobile TCP feed, then you can
program the auxiliary card!
   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
program the auxiliary card!
   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
you can program the auxiliary card! \n   Try to connect the FTP sensor, maybe it
will connect the digital bus!\n   Try to navigate the AGP panel, maybe it will
quantify the mobile alarm!",
  "_time": 1589920332
}

--- output event 2 ---
{
  "_raw": "2020-05-19 16:32:12 moen3628 ipsum[5213]: Use the mobile TCP feed, then
you can program the auxiliary card!\n   Try to connect the FTP sensor, maybe it will
connect the digital bus!\n   Try to navigate the AGP panel, maybe it will quantify
the mobile alarm!",
  "_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:

```
--- input ---
#fields ts      uid     id.orig_h       id.orig_p       id.resp_h       id.resp_p
proto
#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
udp",
  "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
udp",
  "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 slider 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:

```
--- input ---
{"Records":[{"eventVersion":"1.05","eventTime":"2020-04-
08T01:35:55Z","eventSource":"ec2.amazonaws.com","eventName":"DescribeVolumes",
"more_fields":"..."},
{"eventVersion":"1.05","eventTime":"2020-04-
08T01:35:56Z","eventSource":"ec2.amazonaws.com","eventName":"DescribeInstanceAttribute"
"more_fields":"..."}]}

--- output event 1 ---
{
  "_raw": "{\"eventVersion\":\"1.05\",\"eventTime\":\"2020-04-
08T01:35:55Z\",\"eventSource\":\"ec2.amazonaws.com\",\"eventName\":\"DescribeVolumes\",
\"more_fields\":\"...\"}",
  "_time": 1586309755,
  "cribl_breaker": "j-array"
}

--- output event 2 ---
{
  "_raw": "{\"eventVersion\":\"1.05\",\"eventTime\":\"2020-04-
08T01:35:56Z\",\"eventSource\":\"ec2.amazonaws.com\",\"eventName\":\"DescribeInstanceAt
\"more_fields\":\"...\"}",
  "_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:

```
--- input ---
{"time":"2020-05-
25T18:00:54.201Z","cid":"w1","channel":"clustercomm","level":"info","message":"metric
sender","total":720,"dropped":0}
{"time":"2020-05-
25T18:00:54.246Z","cid":"w0","channel":"clustercomm","level":"info","message":"metric
sender","total":720,"dropped":0}


--- output event 1 ---
{
  "_raw": "{\"time\":\"2020-05-
25T18:00:54.201Z\",\"cid\":\"w1\",\"channel\":\"clustercomm\",\"level\":\"info\",\"mess
sender\",\"total\":720,\"dropped\":0}",
  "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\":\"clustercomm\",\"level\":\"info\",\"mess
sender\",\"total\":720,\"dropped\":0}",
  "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:

```
--- input ---
{"level":"debug","ts":"2021-02-
02T10:38:46.365Z","caller":"sdk/sync.go:42","msg":"Handle ENIConfig Add/Update: us-
west-2a, [sg-426fdac8e5c22542], subnet-42658cf14a98b42"}
{"level":"debug","ts":"2021-02-
02T10:38:56.365Z","caller":"sdk/sync.go:42","msg":"Handle ENIConfig Add/Update: us-
west-2a, [sg-426fdac8e5c22542], subnet-42658cf14a98b42"}


--- output event 1 ---
{
  "_raw": "{\"level\":\"debug\",\"ts\":\"2021-02-
02T10:38:46.365Z\",\"caller\":\"sdk/sync.go:42\",\"msg\":\"Handle ENIConfig
Add/Update: us-west-2a, [sg-426fdac8e5c22542], subnet-42658cf14a98b42\"}",
  "_time": 1612262326.365
}

--- output event 2 ---
{
  "_raw": "{\"level\":\"debug\",\"ts\":\"2021-02-
02T10:38:56.365Z\",\"caller\":\"sdk/sync.go:42\",\"msg\":\"Handle ENIConfig
Add/Update: us-west-2a, [sg-426fdac8e5c22542], subnet-42658cf14a98b42\"}",
  "_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:

```
--- 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\",\"43322\",\"0.7
  "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\",\"143433\",\"0.
  "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.

Exporting and Importing 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. Click to open an existing ruleset, or create a new ruleset.
2. In the resulting modal, click **Advanced Mode** to open the JSON editor.
3. You can now modify the ruleset directly in JSON, if you choose.
4. Click **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, click **Advanced Mode** to open the JSON editor.
3. Click **Import**, and choose the file you want.
4. Click **OK** to get back to the **New Ruleset** modal.
5. Click **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: Building 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.

Specifying 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.

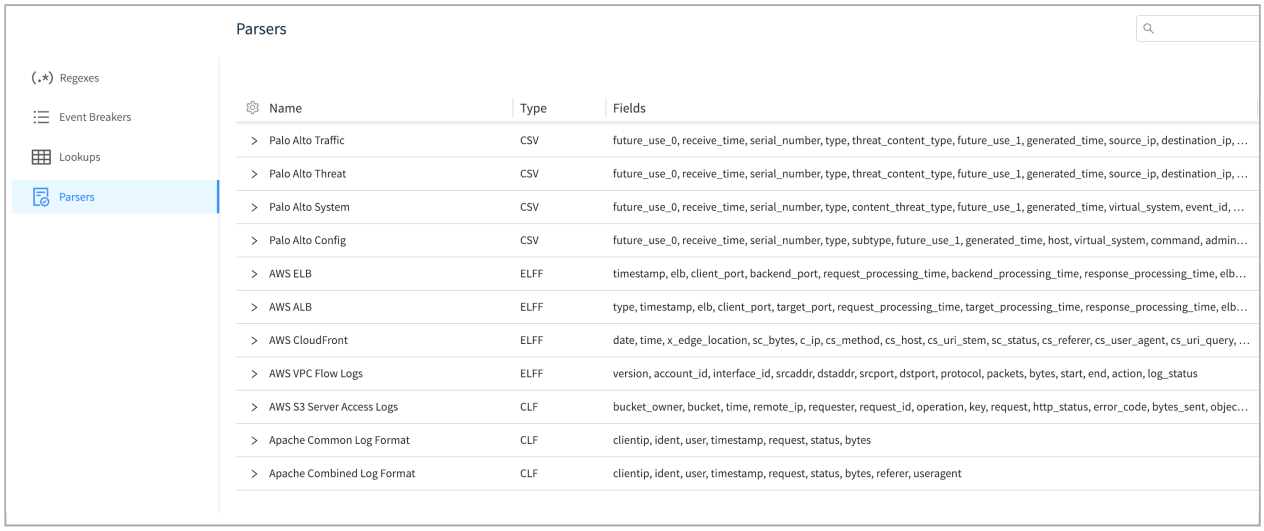
When determining which Event Breaker to use, Cribl Edge creates a substring from the incoming `_raw` field, at the length set by rules' **Max Event Bytes** parameters. If Cribl Edge does not get a match, it checks whether the substring of `_raw` has at least the threshold number of characters set by **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).

13.3. Parsers Library

What Are Parsers

Parsers are definitions and configurations for the Parser Function. You can find the library from Cribl Edge's top nav under **Processing** > **Knowledge** > **Parsers**, and its purpose is to provide an interface for creating and editing Parsers. The library is searchable, and each parser can be tagged as necessary.



Parsers Library

Parsers can be used to **extract** or **reserialize** events. See Parser Function page for examples.

Supported Parser Types:

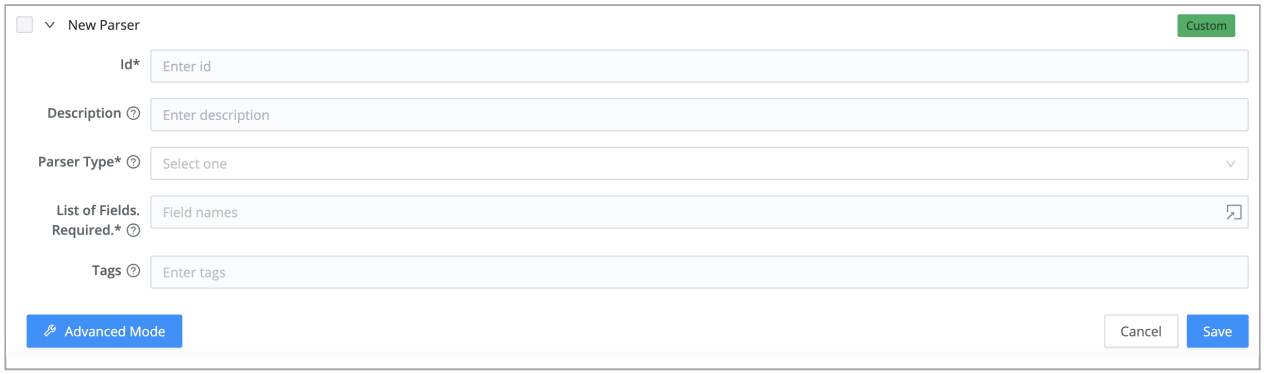
- CSV – Parse and reserialize comma-separated values.
- ELFF – Parse and reserialize events in Extended Log File Format.
- CLF – Parse and reserialize events in Common Log Format.

Creating a Parser

To create a parser, follow these steps:

1. Go to **Knowledge > Parsers** and click **Add New**.
2. Enter a unique **ID**.
3. Optionally, enter a **Description**.

4. Select a **Parser type** (see the supported types above).
5. Enter the **List of fields** expected to be extracted, in order. Click 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.
6. Optionally, enter any desired **Tags**.



Adding a new parser

13.4. Global Variables Library

What Are Global Variables

Global Variables are reusable JavaScript expressions that can be accessed in Functions in any Pipeline. You can access the library from Cribl Edge's top nav under **Processing** > **Knowledge** > **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` <-- ships with Cribl Edge by default.
- 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` <-- ships with Cribl Edge by default.
- Global Variable Value: `Date.now()/1000`
- **Sample Eval Function:** `nowEpoch = C.vars.epoch()`

Scenario 3:

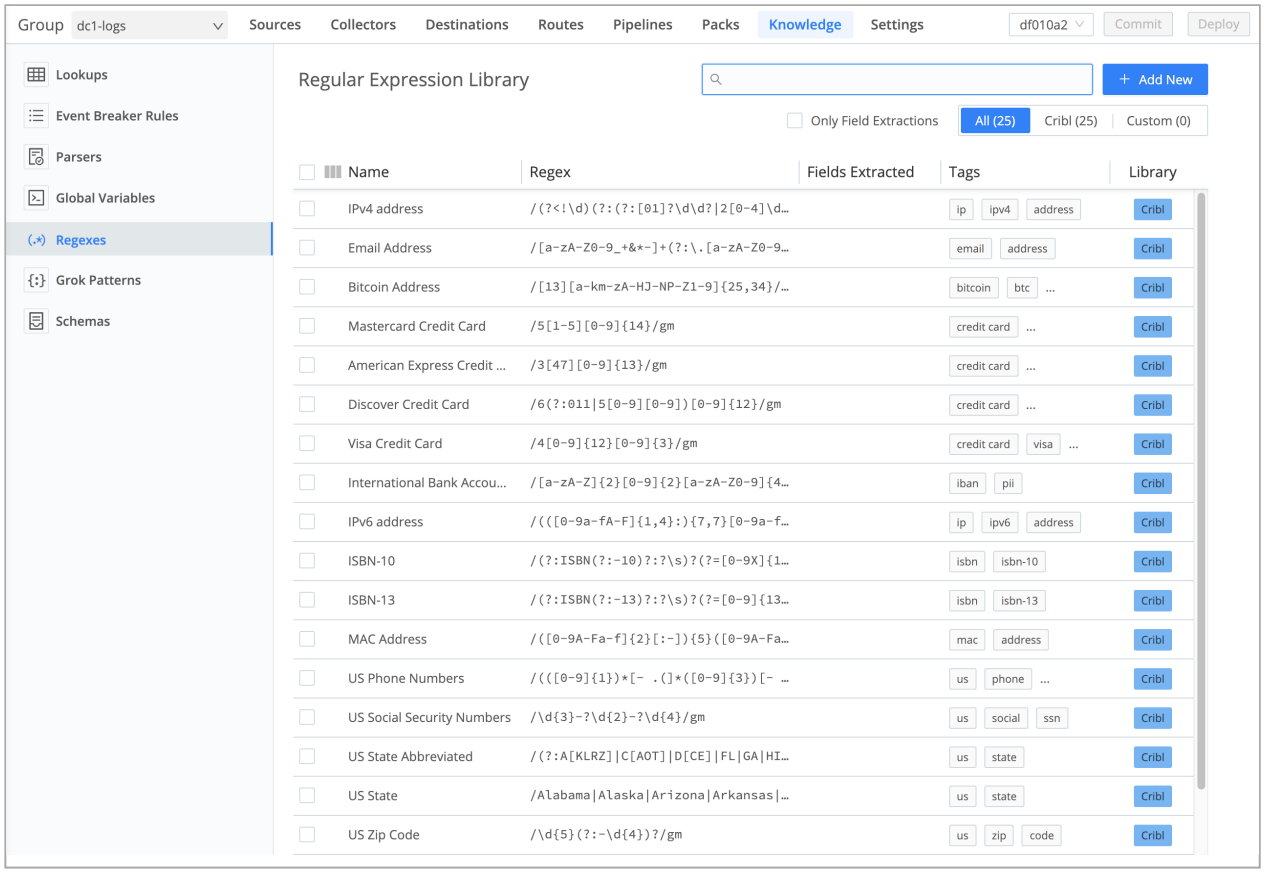
Create a new field called `storage`, by converting the value of event field `size` to human-readable format.

- Global Variable Name: `convertBytes` <-- ships with Cribl Edge by default
- 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.

13.5. Regex Library

What Is the Regex Library

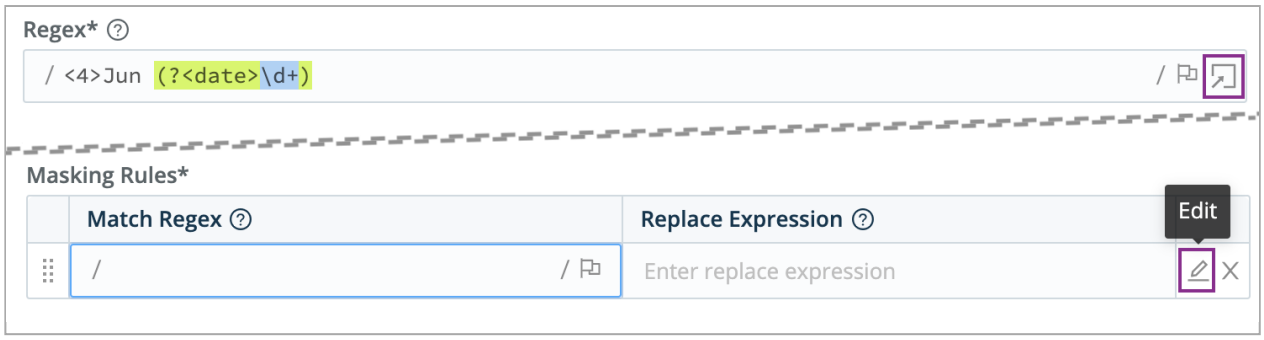
Cribl Edge ships with a Regex Library that contains a set of pre-built common regex patterns. This library serves as an easily accessible repository of regular expressions. The Library is searchable, and you can assign tags to each pattern for further organization or categorization. Access the Library from Cribl Edge's top nav under **Processing** > **Knowledge** > **Regex Library** .



Regular Expression Library

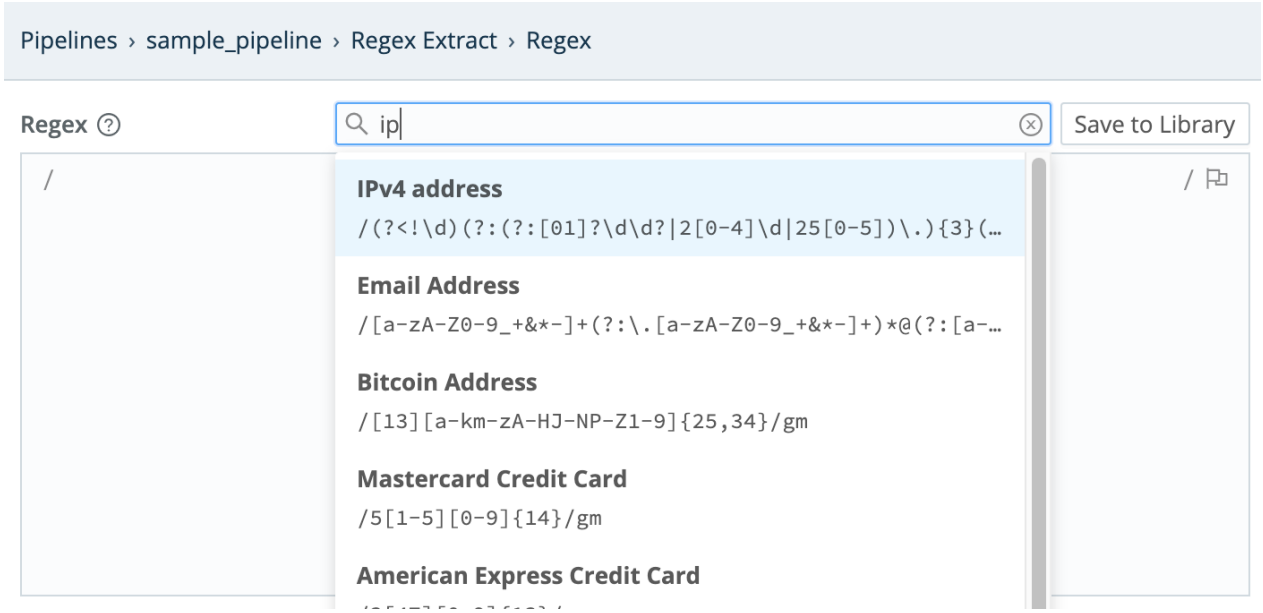
Using Library Patterns

As of this version, the Library contains 25 patterns shipped by Cribl Edge. To insert a pattern into a Function's regex field, first click the pop-out or Edit icon beside that field.



Opening a Regex modal

In the resulting Regex or Rules modal, Regex Library patterns will appear as typeahead options. Click a pattern to paste it in. You can then use the pattern as-is, or modify it as necessary.



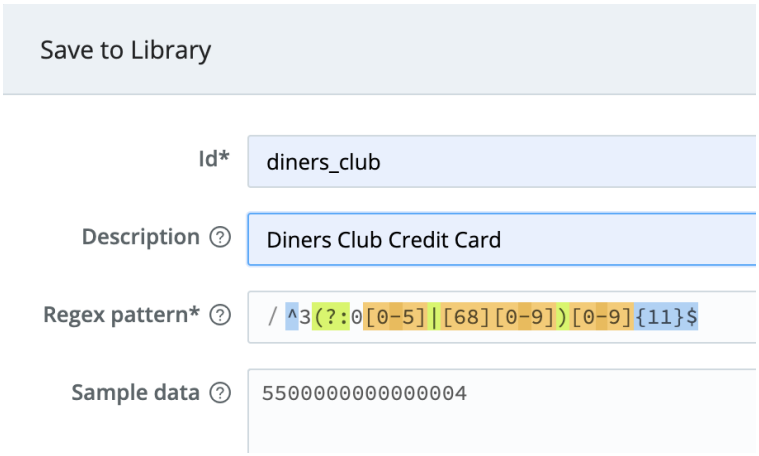
Inserting a pattern from the Regex Library

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, click the **Save to Library** button.

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 click **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

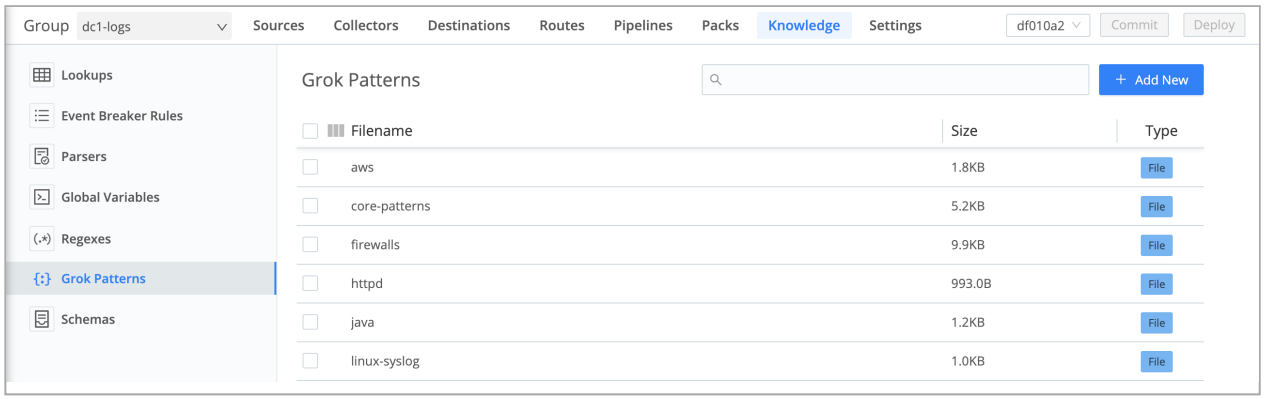
Within the Library, patterns shipped by Cribl will be listed under the **Cribl** tab, while those built by users will be found under **Custom**. Over time, Cribl Edge will ship more patterns, and this distinction allows for both sets to grow independently.

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.

13.6. 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.



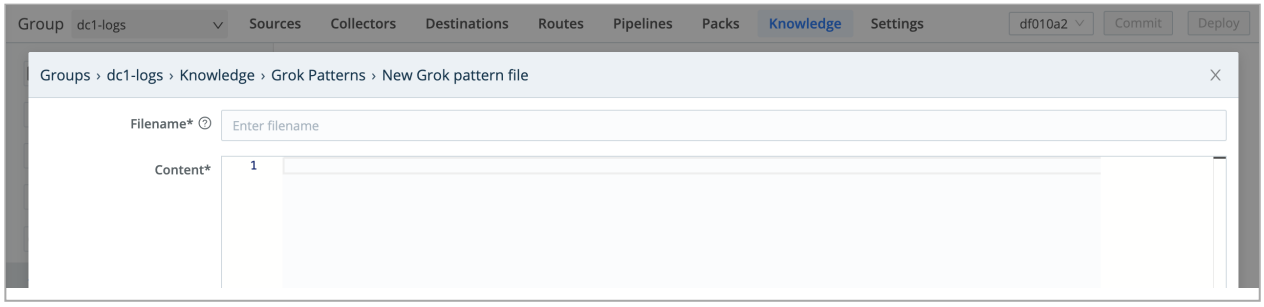
Grok Patterns Library

Managing Library Patterns

You can access the Grok Patterns Library from the UI's top nav by selecting **Processing** > **Knowledge** > **Grok Patterns**. The library contains several pattern files that Cribl provides for basic Grok scenarios, and is searchable.

To edit a pattern file, click **Edit** in its **Actions** column.

To create a new pattern file, click **New Grok pattern file**. In the resulting modal, assign a unique **Filename**, populate the file with patterns, then click **Save**.



Adding Grok patterns

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).

13.7. Schemas Library

Cribl Edge supports two kinds of schemas:

- JSON schemas for validating JSON events. This page outlines how to manage these in the UI at **Knowledge** > **Schemas**.
- 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.

You can access the schema library from Cribl Edge's top nav under **Processing** > **Knowledge** > **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. (E.g., 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, go to **Processing** > **Knowledge** > **Schemas** and click **New 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:

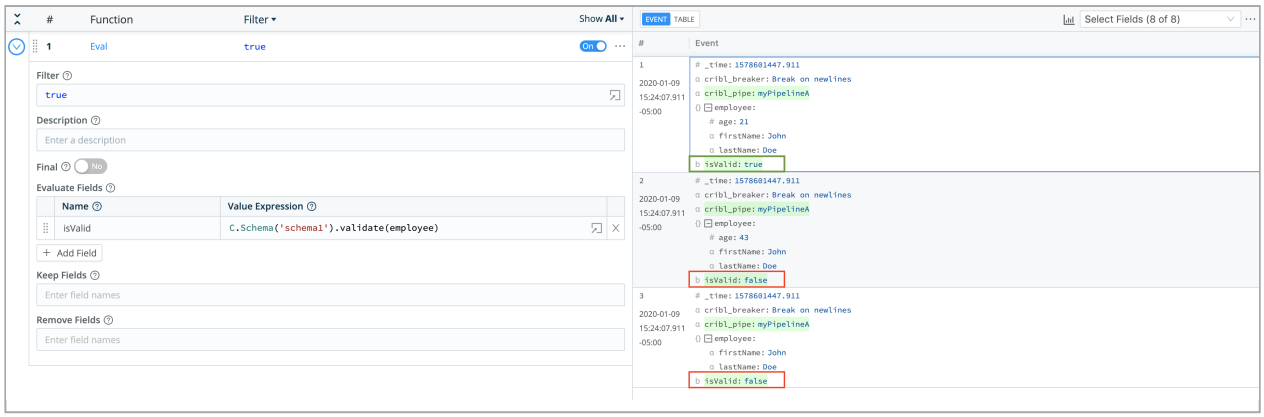
```
{"employee":{"firstName": "John", "lastName": "Doe", "age": 21}}
{"employee":{"firstName": "John", "lastName": "Doe", "age": 43}}
{"employee":{"firstName": "John", "lastName": "Doe"}}
```

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

13.8. Parquet Schemas

Cribl Edge supports two kinds of schemas:

- Parquet schemas for writing data from a Cribl Edge Destination to Parquet files. This page outlines how to manage these in the UI at **Knowledge** > **Parquet Schemas**.
- 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.

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.

From the top nav, select **Processing** > **Knowledge**, then select the **Parquet Schemas** left tab.

If you're adding a Parquet schema:

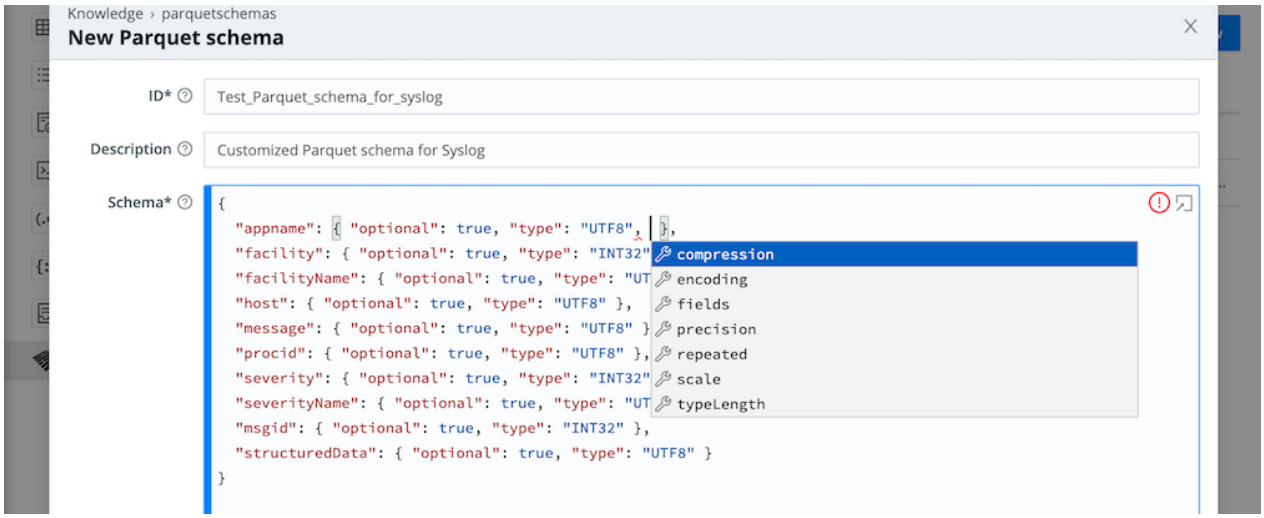
- Click **Add Parquet schema** to open the **New Parquet schema** modal.

If you're cloning an existing Parquet schema to create a new schema:

1. Click the name of the schema you want to start with, to open it in a modal.
2. Click **Clone Parquet Schema** to open the **New Parquet schema** modal.
3. Give the new schema a name and description.

From here, whether you're working with a brand-new or cloned schema:

4. Add and/or edit fields as desired.
5. Click **Save**.



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

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, this ensures that your Destination will now use the newly modified version of the 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 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 (i.e., 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.

13.9. 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.

AppScope is included in your Cribl Edge installation. The `in_appscope` AppScope Source is enabled right out-of-the-box; it is configured to send data to Cribl Edge or Cribl Stream on the same host.

If you need one or more reusable AppScope configurations, create and save them in the AppScope Configs UI described below. They will then be available in the AppScope Config Library.

AppScope Configs ship with a `sample_config` file that contains default settings. You can customize this `sample_config`. Once it's customized, a **Restore Default** button appears at the config modal's lower left; click this if you need to restore the sample file's default settings.

> **Rolling Back a `sample_config` Filter**
>
> When the `sample_config` file has been assigned to an AppScope Source's Rules, you can't restore its default settings. (An error message will state that the AppScope Config file is currently in use.) To resolve this:
>
> On the AppScope Source's **AppScope Filter Settings** tab, delete the `sample_config` from the **Allowlist**. Then, use the AppScope Configs UI to restore the `sample_config`'s default settings. You can now add the refreshed `sample_config` back to the AppScope Source's Allowlist.

Creating AppScope Configs

You access the **AppScope Configs** UI from Cribl Edge's top nav:

- In Cribl Stream, select **Processing** > **Knowledge** > **AppScope Configs**.
- In Cribl Edge, select **More** > **Knowledge** > **AppScope Configs**.

Click **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: Click **Add Protocol** to specify a protocol (e.g., 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, click **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.

Click **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 (i.e., `FOO`). To match when the variable is set with a particular value, enter both (i.e., `FOO=bar`).
- **Ancestor**: Enter a string to match the basename of the scoped process's parent, parent's parent, etc.

Then, in the **Config** column, click the **Edit** button. 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: `^2401000000000000000000000d407`

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`

14. Reference

14.1. 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.

Using 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://logstream.<organizationID>.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.

Obtaining 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 -d '{"username":"
<username","password":"<password>"}'
```

You'll get a JSON object as the response, e.g.:

```
{
    "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

As of Cribl Edge 4.1 (March 2023), Owners and Admins can generate API access tokens directly in their Cribl.Cloud Organization's portal. This is a two-tier process:

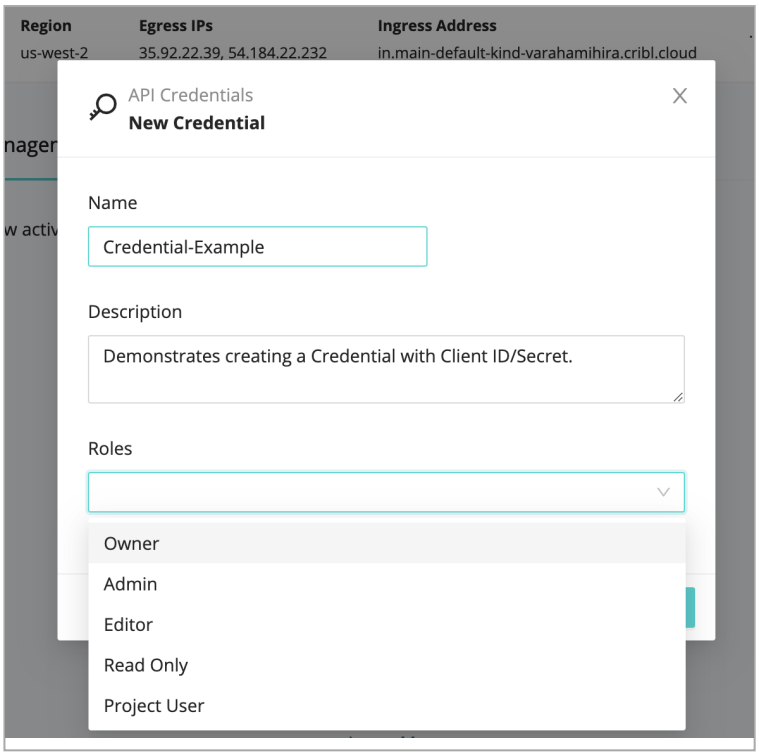
- Generate a Credential to expose a Client ID/Client Secret pair.
- Use the Credential's Client ID and Client Secret to obtain an access token, valid for 24 hours, that can be used to run API calls against Cribl APIs.

Creating an API Credential

Owners and Admins can create Credentials. From your Organization's **Options** (•••) menu, first select **Account** > **Organization**. Then:

1. Select the **API Management** tab.
2. Click **Add Credential**.
3. Specify a **Name**, an optional **Description** of the Credential's purpose, and the **Role** whose permissions the Credential's tokens will grant.

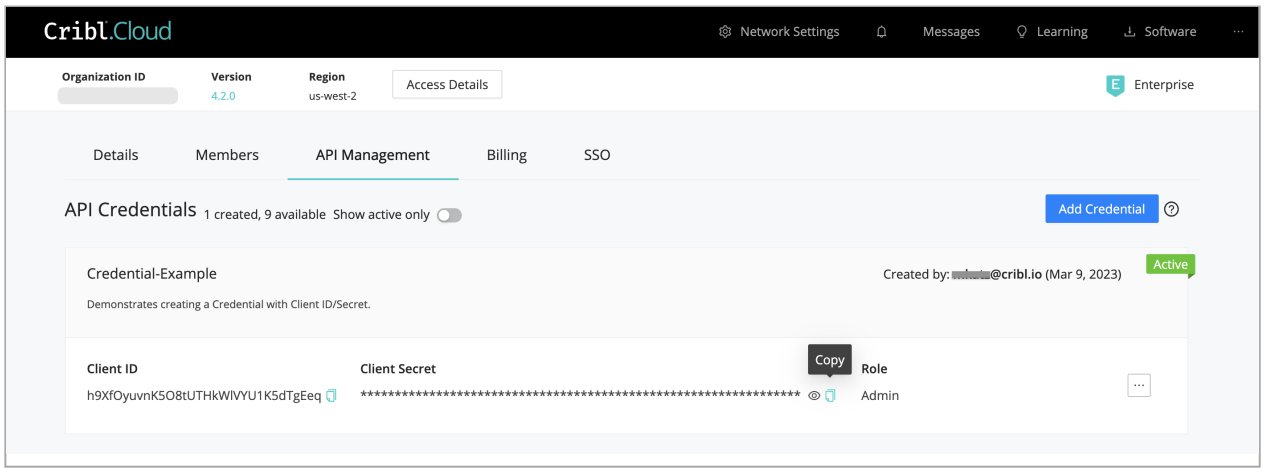
   The **Role** selector is available with an Enterprise plan. Admins can assign Roles up to the Admin level; Owners can assign Roles up to the Owner level. (See Cribl.Cloud Roles.) Without an Enterprise plan, all tokens will be granted the Admin Role.
4. Click **Create** to save the Credential.



Creating a Cribl.Cloud API Credential

Using an API Credential

The new API Credential will appear on the **API Management** page within a few seconds. Each Credential exposes a **Client ID** and **Client Secret**, with Copy buttons. 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

Obtaining 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_secret":"<client_secret>", "audience":"https://api.cribl.cloud"}'
```

Making API Calls

Using the token returned by the preceding example, you can make requests to Cribl APIs. Example curl request, with placeholder token value:

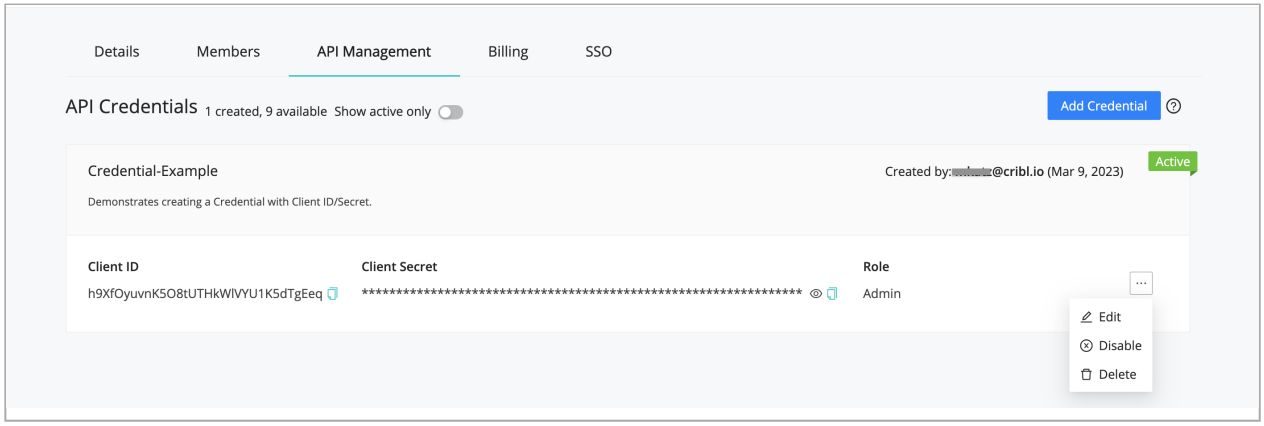
```
curl --location 'https://main-
<organizationID>.cribl.cloud/api/v1/m/default/collectors' \
--header 'Authorization: Bearer <token-generated-above>'
```

Refreshing 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.

Managing API Credentials

A Cribl.Cloud Organization's Owners and Admins can view, edit, and disable existing Credentials. Only Owners can delete Credentials.



Managing Cribl.Cloud access tokens

Update Basic Configurations

You can use the API to programmatically update the configuration of any object type that the API supports (e.g., 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:

   ```
   curl -X GET "<url>/api/v1/system/inputs/<output id>" -H "accept:
   application/json" -H "Authorization: Bearer <token>"
   ```

   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, e.g., `minio_042`.

- Edit the values of the S3 key fields, e.g., `"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 (i.e., 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:

```
curl -X PUT "<url>/api/v1/m/default/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 filename with the response from the previous `PUT` request. This both creates the Lookup and moves the Lookup file to its final destination. Example:

```
curl -X POST "<url>/api/v1/m/default/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 filename in the URL and the body. Example:

```
curl -X PATCH "<url>/api/v1/m/default/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.

Creating 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.

Managing 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 – e.g., in a CI/CD pipeline. For more details, see Managing Packs via API .

14.2. CLI Reference

Command line interface basics

In addition to starting and stopping the Cribl Edge server, Cribl Edge's 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 Cribl Edge's 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: 3.4.0
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
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
pipe               - Feed stdin to a pipeline
vars               - Manage global variables
```

> As of version 3.0, Cribl Edge's former "master" application components are renamed "leader." As long as some legacy terminology remains within CLI commands/options, configuration keys/values, and environment variables, this document will reflect that.

`mode-master`

Configures Cribl Edge as a Leader instance.

Usage

```
./cribl mode-master <options> <args>
```

Options

```
[-H <host>]             - Host (defaults to 0.0.0.0).
[-p <port>]             - Port (defaults to 4200).
[-n <certName>]         – Name of saved certificate. Mutually exclusive with -k or -
c.
[-k <privKeyPath>]      – Path on server to the private key to use. PEM format. Can
reference $ENV_VARS. Requires -c, to successfully connect to the Leader. Mutually
exclusive with -n.
[-c <certPath>]         – Path on server to the certificate to use. PEM format. Can
reference $ENV_VARS. Requires -k, to successfully connect to the Leader. Mutually
exclusive with -n.
[-u <authToken>]        - Optional authentication token to include as part of the
connection header.
[-i <ipWhitelistRegex>] – Regex matching IP addresses that are allowed to establish
a connection.
[-r <resiliency>]       – Resiliency mode for the Leader. Accepts agruments: none,
failover.
[-v <failoverVolume>]   – If the -r option is set to failover, this defines the
volume to use as shared storage.
```

Sample Response

```
Settings updated.
You will need to restart Cribl Stream before your changes take full effect.
```

mode-single

Configures Cribl Stream as a single-instance deployment.

Usage

```
./cribl mode-single [--help]
```

Sample Response

```
Settings updated.
You will need to restart Cribl Stream before your changes take full effect.
```

mode-edge

Configures Cribl Edge as a single-instance deployment.

Usage

```
./cribl mode-edge [--help]
```

Sample Response

```
Settings updated.
```

mode-worker

Configures Cribl Stream as a Worker instance.

Usage

```
./cribl mode-worker -H <host> -p <port> <options> <args>
```

The `-H <host> -p <port>` parameters are required.

Options

```
-H <host>          – Leader Node's Hostname or IP address.
-p <port>          – Leader Node's cluster communications port (defaults to 4200).
[-S <true|false>]  – Sets, or disables, secure communication between Leader and
Worker Nodes via TLS. UI-generated version-update scripts automatically set this
option according to the Leader's configuration.
[-n <certName>]    – Name of saved certificate. Mutually exclusive with -k or -c.
[-k <privKeyPath>] – Path on server to the private key to use. PEM format. Can
reference $ENV_VARS. Requires -c, to successfully connect to the Leader. Mutually
exclusive with -n.
[-c <certPath>]    – Path on server to the certificate to use. PEM format. Can
reference $ENV_VARS. Requires -k, to successfully connect to the Leader. Mutually
exclusive with -n.
[-u <authToken>]   – Authentication token to include as part of the connection
header. By default, this token is included and is set to 'criblmaster'.
[-e <envRegex>]    – Regex that selects environment variables to report to Leader.
Requires slashes to pass validation, e.g., /test/
[-t <tags>]        – Tag values to report to Leader.
[-g <group>]       – Worker Group/Fleet to report to Leader.
```

Sample Response

```
Settings updated.
You will need to restart Cribl Stream before your changes take full effect.
```

mode-managed-edge

Configures Cribl Edge as an Edge Node.

Usage

```
./cribl mode-managed-edge -H <host> -p <port> <options> <args>
```

The `-H <host> -p <port>` parameters are required.

Options

```
-H <host>          – Leader Node's Hostname or IP address.
-p <port>          – Leader Node's cluster communications port (defaults to 4200).
[-S <true|false>]  – Sets, or disables, secure communication between Leader and
Worker Nodes via TLS. UI-generated version-update scripts automatically set this
option according to the Leader's configuration.
[-n <certName>]    – Name of saved certificate. Mutually exclusive with -k or -c.
[-k <privKeyPath>] – Path on server to the private key to use. PEM format. Can
reference $ENV_VARS. Requires -c, to successfully connect to the Leader. Mutually
exclusive with -n.
[-c <certPath>]    – Path on server to the certificate to use. PEM format. Can
reference $ENV_VARS. Requires -k, to successfully connect to the Leader. Mutually
exclusive with -n.
[-u <authToken>]   – Authentication token to include as part of the connection
header. By default, this token is included and is set to 'criblmaster'.
[-e <envRegex>]    – Regex that selects environment variables to report to Leader.
Requires slashes to pass validation, e.g., /test/
[-t <tags>]        – Tag values to report to Leader.
[-g <group>]       – Worker Group/Fleet to report to Leader.
```

Sample Response

```
Settings updated.
```

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.
  [-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
source
-------------------------------------------------------------------------------
----------------------------------------
HelloPacks  1.0.0    ----  Hello, Packs!  Cribl, Inc.  A sample pack with a simple
example  file:/opt/cribl/default/HelloPacks
```

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 18
............
Cribl Stream is not running
Starting Cribl Stream...
...
Cribl Stream started
```

start

Starts Cribl Edge. Executes immediately. Upon first run, echoes Cribl Edge's default login credentials.

Usage

```
./cribl start <options> <args>
```

Options

```
[-d <dir>]  - Configuration directory
[-r <role>] - Process role
```

Sample Response

```
Starting Cribl Stream...
...
Cribl Stream 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

```
Cribl Stream Status

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 Stream, process 3951
............
Cribl Stream is not running
```

`version`

Displays Cribl Edge version. Executes immediately.

Usage

```
./cribl version [--help]
```

Sample Response

```
Software Version: 3.1.0-f765e418
```

auth

Log into or out of Cribl Edge.

Usage

```
./cribl auth <sub-command> <options> <args>
```

Sub-commands and Options

```
login             - Log in to Cribl Stream/Edge, args:
  [-h <oldHost>]  - undefined
  [-H <host>]     - Host URL (e.g. http://localhost:9000)
  [-u <username>] - Username
  [-p <password>] - Password
  [-f <file>]     - File with credentials
logout            - Log out from Cribl Stream/Edge
```

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 API host and port are listed in the `cribl.yml` file's `api:` section.

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 Cribl Stream/Edge boot-start, args:
  [-m <manager>]    - Init manager (systemd|initd)
  [-c <configDir>]  - Config directory for the init manager
enable              - Enable Cribl Stream/Edge boot-start, args:
  [-m <manager>]    - Init manager (systemd|initd)
  [-u <user>]       - User to run Cribl Stream/Edge as
  [-c <configDir>]  - Config directory for the init manager
```

Sample Response

```
Enabling Cribl Stream/Edge to be managed by initd...
boot-start enable command needs root privileges...
Enabled Cribl Stream/Edge to be managed by initd as user=root.
```

diag

Manages diagnostic bundles.

Usage

```
./cribl diag <sub-command> <options> <args>
```

Sub-commands and Options

```
create              - Creates diagnostic bundle for Cribl Stream/Edge, args:
  [-d ]             - Run create in debug mode
  [-j ]             - Do not append '.txt' to js files
  [-t <number> ]    - Latest number of jobs to include in bundle
  [-M ]             - Exclude metrics from bundle to reduce file size
list                - List existing Cribl Stream/Edge diagnostic bundles

send                - Send Cribl Stream/Edge diagnostics bundle to Cribl Support,
args:
   -c <caseNumber>  - Cribl Support Case Number
  [-p <path>]       - Diagnostic bundle path (if empty then new bundle will be
created)

heapsnapshot        - Creates a heap snapshot for Cribl Stream/Edge processes, args:
   -p <pid>         - Process ID
```

Sample Responses

diag create

```
Created @[product] diagnostic bundle at /opt/cribl/diag/logstream-zedborcdb72f-
20210820T204405.tar.gz
```

diag heapsnapshot -p 12345

The response format is `Heap-<epoch-timestamp>-<pid>.heapsnapshot`:

```
Heap-1672574400000-12345.heapsnapshot
```

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 & 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`, e.g.: `./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),
aes-256-gcm.
[-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.
 -g <group>       - Worker Group's ID.
list              - List encryption keys, args:
 -g <group>       - Worker Group's 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 6]               - Listen on IPv6 addresses ("::" format). Omit this option — or
use "-f 4" — to listen on the default IPv4 addresses ("0.0.0.0").
[-s <statsInterval>] - Stats output interval (ms), use 0 to disable.
[-u]                 - Listen on UDP port instead.
[-o]                 - Output received data to stdout.
[-t <throttle>]      - throttle rate in (unit)/sec, where units can be KB,MB,GB, and
TB.
```

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/filename 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
[-a <pack>]       - Optional Cribl Pack context
```

Sample Response

```
...
{"time":"2021-08-
20T20:37:00.017Z","cid":"api","channel":"commands","level":"info","message":"creating
new pipeline","id":"main","conf":{"asyncFuncTimeout":1000,"functions":
[{"id":"eval","disabled":false,"filter":"true","conf":{"add":
[{"name":"cribl","value":"'yes'"}],"remove":[]}}]}}
{"time":"2021-08-
20T20:37:00.019Z","cid":"api","channel":"pipe:main","level":"info","message":"start
loading and initializing functions","count":1}
{"time":"2021-08-
20T20:37:00.021Z","cid":"api","channel":"pipe:main","level":"info","message":"finished
loading and initializing functions","count":1}
{"time":"2021-08-
20T20:37:00.022Z","cid":"api","channel":"commands","level":"info","message":"START
pushing stdin events","id":"main"}
{"time":"2021-08-
20T20:37:00.028Z","cid":"api","channel":"GrokMgr","level":"info","message":"loaded
grok patterns","count":152}
...
```

`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"
  }
]
```

14.3. 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`
Regexes Configuration	`(default|local)/cribl/regexes.yml`

CATEGORY	RELATIVE PATH
Breakers Configuration	`(default|local)/cribl/breakers.yml`
Limits Configuration	`(default|local)/cribl/limits.yml`
Pipelines Configuration	`(default|local)/cribl/pipelines/<pname>` Each Pipeline's conf is contained therein.
Routes Configuration	`(default|local)/cribl/pipelines/routes.yml`
Functions	`(default|local)/cribl/functions/<function_name>` Each function's code, conf is contained therein.
Functions Configuration	`(default|local)/cribl/functions/<function_name>/...` Each function's conf is contained therein.
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.

Configurations and Restart

> You can **Restart** and **Reload** via the UI. On the top nav, go to **Settings** > **Global Settings** > **System** > **Controls** then click on the **Reload** button or the **Restart** button.

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.

On restarts, be aware of the following:

- Data in process of being received on Syslog/UDP might be dropped.
- Edge Nodes will temporarily disappear from the Leader's Manage/Edge Node page.
- Aggregations and suppressions will start over.
- The local copy of monitoring metrics on the Edge Nodes, before the restart, will be dropped.
- Any events in RAM for persistent queue (PQ) will be lost. PQ data on disk will persist.

Changes that require reloads include configuration changes to:

- Functions

- Pipelines
- Routes
- Lookups
- Parquet schemas
- Global variables
- Group Settings: Limits and Logging Levels

Changes that require restarts include configuration changes to:

- Mode of operation (Leader, distributed Edge Node, Single-instance)
- Fleet assignment
- Event Breakers
- Quick Connects
- Sources
- Destinations
- Group Settings: Advance Settings, TLS
- Group Settings: Worker Processes - Process Count/Memory

Some general guidelines to keep in mind:

- Configuration changes resulting from 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**. You will be prompted to confirm 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** 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

Similar to 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`.

14.3.1. cribl.yml

`cribl.yml` contains settings for configuring API and other system properties.

$CRIBL_HOME/default/cribl/cribl.yml

```
api:
  # Address to bind to. Default: 0.0.0.0
  host: 0.0.0.0
  # Port to listen to. Default: 9000
  port: 9000
  # Number of times to retry binding to API port.
  retryCount: 120
   # Period, in seconds, between consecutive port binding retries. Default: 120.
  retrySleepSecs: 5
  # URL base path from which to serve all assets (useful when behind a proxy).
Default: 5.
  baseUrl: ""
  # Flag to enable/disable local UI access. Default: false
  disabled : false
  # Enable teleporting to @{node}s'. Default: false
  workerRemoteAccess: false
  # Log users out when their roles change. Default: true
  revokeOnRoleChange: true
  # How long (in seconds) authentication tokens remain valid. Default is 1 hr.;
minimum is 1 sec.'
  authTokenTTL: 60
  # How long (in seconds) @{product} will observe no user interaction before
invalidating user's session tokens. Default is 1 hr.; minimum is 60 sec.
  idleSessionTTL: true
  # Custom HTTP headers to be sent with every response.
  headers: true
  # Rate limit, expressed as maximum number of requests per interval (second,
minute, hour, day). E.g., 3/second, 5/minute.
  loginRateLimit: 2/second
  # Rate limit for SSO and SLO callback endpoints. Expressed as maximum number of
requests per second, minute, hour, or day, e.g., 3/second, 5/minute. When limit is
reached, the @{product} sends 429 Too Many Requests HTTP responses.
  ssoRateLimit: 3/second
  # Toggle to 'No' to disable caching of browser's frequent API requests. (Can slow
the UI's response time.)
  apiCache: 3/second
  # SSL Settings
  ssl:
    # SSL is enabled by default
    disabled: false
    # Path to private key
    privKeyPath: /path/to/privkey.pem
    # Path to certificate
    certPath: /path/to/cert.pem

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://),
disable for unsecure (ldap://) connection.
  ldapServers: # [array of strings] LDAP servers - List of LDAP servers, each entry
should contain host:port (e.g., localhost:389)
  bindDN: # [string] Bind DN - Distinguished name of entity to authenticate with
LDAP server, e.g., 'cn=admin,dc=example,dc=org'
  bindCredentials: # [string] Password - Distinguished Name password used to
authenticate with LDAP server
  searchBase: # [string] User search base - Starting point to search LDAP for users,
```

```
e.g., 'dc=example,dc=org'
  usernameField: # [string] Username field - LDAP user search field, e.g. cn or uid
  searchFilter: # [string] User search filter - LDAP search filter to apply when
finding user, e.g. (&(group=admin)(!(department=123*)))
  groupSearchBase: # [string] Group search base - Starting point to search LDAP for
groups, e.g., 'dc=example,dc=org'
  groupMemberField: # [string] Group member field - LDAP group search field, e.g.
member
  groupSearchFilter: # [string] Group search filter - LDAP search filter to apply
when finding group, e.g. (&(cn=cribl*)(objectclass=group))
  groupField: # [string] Group name field - LDAP group field, e.g. cn
  connectTimeout: # [number] Connection timeout (ms)
  rejectUnauthorized: # [boolean] Reject unauthorized - Valid for secure LDAP
connections, set true to reject unauthorized server certificates.
  fallback: # [boolean] Fallback on fatal error - Attempt local authentication if
LDAP authentication is down or mis-configured. Defaults to false.
  groups: # [object]
    default: # [string] Default role - Default role assigned to groups not
explicitly mapped to a role.
    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
instance is running.
  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
authentication. Defaults to localhost.
  port: # [number] Port - Port of Splunk instance that provides authentication.
Defaults to 8089.
  ssl: # [boolean] SSL - Whether SSL is enabled on Splunk instance that provides
authentication. Defaults to Yes.
  fallback: # [boolean] Fallback on fatal error - Attempt local authentication if
Splunk is unsuccessful. Defaults to false.
  groups: # [object]
    default: # [string] Default role - Default role assigned to groups not
explicitly mapped to a role.
    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
entries are also allowed.
  audience: # [string] Audience - The Audience from provider configuration. This
will be the base URL of your Cribl server, i.e.: https://yourDomain.com:9000
  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:
openid profile email.
  auth_url: # [string] Authentication URL - The full path to the provider's
authentication endpoint. Be sure to configure the callback URL at provider as
<yourDomainUrl>/api/v1/auth/authorization-code/callback, e.g.
https://yourDomain.com:9000/api/v1/auth/authorization-code/callback
  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
url. If not provided, Stream will attempt to gather user info from the ID token
returned from the Token URL.
  logout_url: # [string] Logout URL - The full path to the provider's logout URL.
Leave blank if the provider does not support logout or token revocation.
  userIdExpr: # [string] User identifier - Expression used to derive userId from the
id_token returned by the openId provider.
  rejectUnauthorized: # [boolean] Validate certs - Validate certificates; set to
false to allow insecure self-signed certificates.
  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
user groups.
  fallback: # [boolean] Allow local auth - Allows locally configured users to log in
to Stream.
  groups: # [object]
    default: # [string] Default role - Default role assigned to groups not
explicitly mapped to a role.
    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
TLSv1.3.
  defaultCipherList: # [string] Default cipher list - Default suite of enabled and
disabled TLS ciphers. Defaults to:
      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 of curve NIDs or names, to use for ECDH key agreement. For example:
'Pâ□□521:Pâ□□384:Pâ□□256'. Defaults to 'auto'.
  rejectUnauthorized: # [boolean] Validate server certs - Whether to validate server
certificates globally. Set to false to allow self-signed certs.
proxy: # [object]
  useEnvVars: # [boolean]
git: # [object]
  branch: # [string] Branch - The branch to track in your Stream deployment's git
repository.
  gitOps: # [string] GitOps workflow - The GitOps workflow for managing Stream's
config.
  commitDeploySingleAction: # [boolean] Collapse actions - When enabled, Commit &
Deploy will be collapsed into a single action. If you've configured a remote, Commit
& Git Push will also be collapsed. Your DefaultÂ CommitÂ Message below will be used
for all commits.
  defaultCommitMessage: # [string] Default commit message - Enter a default message
to use for all commits.
  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
use for authentication on remote git repo.
  strictHostKeyChecking: # [boolean] SSH strict host key checking - Validate key
against known hosts, to prevent spoofing or impersonation attacks. For details, see
"VerifyingÂ Host Keys" [here](https://linux.die.net/man/1/ssh).

  # -------------------------------------------------------


  # -------------- if authType is basic ---------------

  user: # [string] User - Username for authentication.
  password: # [string] Password - Password for authentication. (With GitHub, use a
personal access token.)

  # -------------------------------------------------------

  autoAction: # [string] Scheduled global actions - Global git actions to run
automatically on a schedule.

  # -------------- if autoAction is commit ---------------

  autoActionSchedule: # [string] Schedule - Cron schedule to run selected git action
on.
  autoActionMessage: # [string] Commit message - Default scheduled commit message.

  # -------------------------------------------------------


  # -------------- if autoAction is push ---------------
```

```
  autoActionSchedule: # [string] Schedule - Cron schedule to run selected git action
on.

  # -------------------------------------------------------

  timeout: # [number] Git timeout - Max time (milliseconds) to wait for git
processes before killing them, 0 to wait indefinitely
```

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
```

14.3.2. 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
data to apply rule to. To test your sample, use the maximize icon on the right.
      type: # [string] Event Breaker Type - Event Breaker Type
      timestampAnchorRegex: # [string] Timestamp Anchor - Regex to match before
attempting timestamp extraction. Use $ (end of string anchor) to not perform
extraction.
      timestamp: # [object] Timestamp Format - Auto, manual format (strptime) or
current time.
      type: # [string] Timestamp Type
      length: # [number] Length
      format: # [string] Format
      timestampTimezone: # [string] Default timezone - Timezone to assign to
timestamps without timezone info.
      timestampEarliest: # [string] Earliest timestamp allowed - The earliest
timestamp value allowed relative to now. E.g., -42years. Parsed values prior to this
date will be set to current time.
      timestampLatest: # [string] Future timestamp allowed - The latest timestamp
value allowed relative to now. E.g., +42days. Parsed values after this date will be
set to current time.
      maxEventBytes: # [number] Max Event Bytes - The maximum number of bytes that
an event can be before being flushed to the pipelines
      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
value (can be constant).
      disabled: # [boolean] Disabled - Allows breaker rule to be enabled or
disabled, default is enabled.
```

14.3.3. certificates.yml

`certificates.yml` maintains a list of configured certificates and their parameters.

$CRIBL_HOME/local/cribl/certificates.yml

```
cerficate_id: # [object]
  description: # [string] Description - Brief description of this certificate.
Optional.
  cert: # [string] Certificate - Drag/drop or upload host certificate, in PEM/Base64
format. Or paste its contents here.
  privKey: # [string] Private key - Certificate private key.
  passphrase: # [string] Passphrase - Passphrase. Optional.
  ca: # [string] CA certificate - Optionally, drag/drop or upload all CA
certificate(s) in PEM/Base64 format. Or paste certs' contents here. Certs can be
used for client and/or server auth.
  inUse: # [array of strings] Referenced - List of configurations referencing this
certificate.
```

14.3.4. 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
for this group
  onPrem: # [boolean] On prem - Whether the group accepts on-prem or cloud worker
nodes (this field is only available on cloud deployments)
  isFleet: # [boolean] Is Fleet - Fleet groups only manage edge nodes.
  workerRemoteAccess: # [boolean] UI access - Enable authenticated viewing of
Workers' UI from the Leader.
```

14.3.5. 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
breaking rulesets that will be applied, in order, to the input data stream.
    staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of
time (in milliseconds) the Event Breaker will wait for new data to be sent to a
specific channel, before flushing the data stream out, as-is, to the Pipelines.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
process its output (stdout)
      args: # [array of strings] Arguments - Arguments
    throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to
throttle while writing to an output. Also takes values with multiple-byte units,
such as KB, MB, GB, etc. (E.g., 42 MB.) Default value of 0 specifies no throttling.
    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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
id>.
      compress: # [string] Compression - Codec to use to compress the persisted
```

```
data.

    # ------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  kafka_input: # [object]
    type: # [string] Input Type
    brokers: # [array of strings] Brokers - List of Kafka brokers to use, eg.
localhost:9092
    topics: # [array of strings] [required] Topic - Topic to subscribe to. Warning:
To optimize performance, Cribl suggests subscribing each Kafka Source to only a
single topic.
    groupId: # [string] Group ID - Specifies the consumer group this instance
belongs to, default is 'Cribl'.
    fromBeginning: # [boolean] From beginning - Whether to start reading from
earliest available data, relevant only during initial subscription.
    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
Confluent Schema Registry, i.e: http://localhost:8081
      tls: # [object] TLS settings (client side)
        disabled: # [boolean] Disabled

        # -------------- if disabled is false ---------------

        rejectUnauthorized: # [boolean] Validate server certs - Reject certs that
are not authorized by a CA in the CA certificate path, or by another trusted CA
(e.g., the system's CA). Defaults to No.
        servername: # [string] Server name (SNI) - Server name for the SNI (Server
Name Indication) TLS extension. It must be a host name, and not an IP address.
        certificateName: # [string] Certificate name - The name of the predefined
certificate.
        caPath: # [string] CA certificate path - Path on client in which to find CA
certificates to verify the server's cert. PEM format. Can reference $ENV_VARS.
        privKeyPath: # [string] Private key path (mutual auth) - Path on client in
which to find the private key to use. PEM format. Can reference $ENV_VARS.
        certPath: # [string] Certificate path (mutual auth) - Path on client in
which to find certificates to use. PEM format. Can reference $ENV_VARS.
        passphrase: # [string] Passphrase - Passphrase to use to decrypt private
key.
        minVersion: # [string] Minimum TLS version - Minimum TLS version to use when
connecting
        maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when
connecting

        # ----------------------------------------------------------


      # ------------------------------------------------------------

    connectionTimeout: # [number] Connection timeout (ms) - Maximum time to wait for
a successful connection.
    requestTimeout: # [number] Request timeout (ms) - Maximum time to wait for a
successful request.
    sasl: # [object] Authentication - Authentication parameters to use when
connecting to brokers. Using TLS is highly recommended.
```

```
      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
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to No.
      servername: # [string] Server name (SNI) - Server name for the SNI (Server
Name Indication) TLS extension. It must be a host name, and not an IP address.
      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      caPath: # [string] CA certificate path - Path on client in which to find CA
certificates to verify the server's cert. PEM format. Can reference $ENV_VARS.
      privKeyPath: # [string] Private key path (mutual auth) - Path on client in
which to find the private key to use. PEM format. Can reference $ENV_VARS.
      certPath: # [string] Certificate path (mutual auth) - Path on client in which
to find certificates to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to use when
connecting
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when
connecting

      # ----------------------------------------------------

    sessionTimeout: # [number] Session timeout (ms) -
      Timeout used to detect client failures when using Kafka's group management
facilities.
      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
rebalance.
      Value must be between the broker's configured group.min.session.timeout.ms and
group.max.session.timeout.ms.
      See details [here]
(https://kafka.apache.org/documentation/#consumerconfigs_session.timeout.ms).
    rebalanceTimeout: # [number] Rebalance timeout (ms) -
      Maximum allowed time for each worker to join the group after a rebalance has
begun.
      If the timeout is exceeded, the coordinator broker will remove the worker from
the group.
      See details [here]
(https://kafka.apache.org/documentation/#connectconfigs_rebalance.timeout.ms).
    heartbeatInterval: # [number] Heartbeat interval (ms) -
      Expected time between heartbeats to the consumer coordinator when using
Kafka's group management facilities.
      Value must be lower than sessionTimeout, and typically should not exceed 1/3
of the sessionTimeout value.
      See details [here]
(https://kafka.apache.org/documentation/#consumerconfigs_heartbeat.interval.ms).
    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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
id>.
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # --------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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
any client (Authorization: <token>). If empty, unauthed access is permitted.
    tls: # [object] TLS settings (server side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------
```

```
      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      privKeyPath: # [string] Private key path - Path on server containing the
private key to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      certPath: # [string] Certificate path - Path on server containing certificates
to use. PEM format. Can reference $ENV_VARS.
      caPath: # [string] CA certificate path - Path on server containing CA
certificates to use. PEM format. Can reference $ENV_VARS.
      requestCert: # [boolean] Authenticate client (mutual auth) - Whether to
require clients to present their certificates. Used to perform client authentication
using SSL certs.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to accept
from connections.
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept
from connections.

      # ------------------------------------------------------

    maxActiveReq: # [number] Max active requests - Maximum number of active requests
per Worker Process. Use 0 for unlimited.
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
is proxied by a device that supports Proxy Protocol V1 or V2.
    captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add
request headers to events, in the __headers field.
    activityLogSampleRate: # [number] Activity log sample rate - How often request
activity is logged at the `info` level. A value of 1 would log every request, 10
every 10th request, etc.
    requestTimeout: # [number] Request timeout (seconds) - How long to wait for an
incoming request to complete before aborting it. Use 0 to disable.
    criblAPI: # [string] Cribl HTTP Event API - Absolute path on which to listen for
the Cribl HTTP API requests. At the moment, only _bulk (default /cribl/_bulk) is
available. Use empty string to disable.
    elasticAPI: # [string] Elasticsearch API endpoint (Bulk API) - Absolute path on
which to listen for the Elasticsearch API requests. At the moment only _bulk
(default /elastic/_bulk) is available. Use empty string to disable
    splunkHecAPI: # [string] Splunk HEC Endpoint - Absolute path on which listen for
the Splunk HTTP Event Collector API requests. Use empty string to disable.
    splunkHecAcks: # [boolean] Splunk HEC Acks - Whether to enable Splunk HEC
acknowledgements
    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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
specified Git branch. If empty, will be enabled everywhere.
```

```
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
id>.
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # ------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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
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
certificate.
      privKeyPath: # [string] Private key path - Path on server containing the
private key to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      certPath: # [string] Certificate path - Path on server containing certificates
to use. PEM format. Can reference $ENV_VARS.
      caPath: # [string] CA certificate path - Path on server containing CA
certificates to use. PEM format. Can reference $ENV_VARS.
      requestCert: # [boolean] Authenticate client (mutual auth) - Whether to
require clients to present their certificates. Used to perform client authentication
using SSL certs.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to accept
from connections.
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept
from connections.

      # ------------------------------------------------------

    ipWhitelistRegex: # [string] IP Allowlist Regex - Regex matching IP addresses
that are allowed to establish a connection.
```

```
    maxActiveCxn: # [number] Max Active Connections - Maximum number of active
connections allowed per Worker Process, use 0 for unlimited
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
is proxied by a device that supports Proxy Protocol V1 or V2.
    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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event
breaking rulesets that will be applied, in order, to the input data stream.
    staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of
time (in milliseconds) the Event Breaker will wait for new data to be sent to a
specific channel, before flushing the data stream out, as-is, to the Pipelines.
    authTokens: # [array] Auth tokens - Shared secrets to be provided by any Splunk
forwarder. IfÂ empty, unauthed access is permitted.
      - token: # [string] Token - Shared secrets to be provided by any Splunk
forwarder. IfÂ empty, unauthed access is permitted.
        description: # [string] Description - Optional token description
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
id>.
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # -------------------------------------------------------
```

```
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  splunk_search_input: # [object]
    searchHead: # [string] Search head - Search head base URL, can be expression,
default is https://localhost:8089.
    search: # [string] [required] Search - Enter Splunk search here. For example:
'index=myAppLogs level=error channel=myApp' OR '| mstats avg(myStat) as myStat WHERE
index=myStatsIndex.'
    earliest: # [string] Earliest - The earliest time boundary for the search. Can
be an exact or relative time. For example: '2022-01-14T12:00:00Z' or '-16m@m'
    latest: # [string] Latest - The latest time boundary for the search. Can be an
exact or relative time. For example: '2022-01-14T12:00:00Z' or '-1m@m'
    cronSchedule: # [string] [required] Cron schedule - A cron schedule on which to
run this job.
    endpoint: # [string] [required] Search endpoint - REST API used to create a
search.
    outputMode: # [string] [required] Output mode - Format of the returned output
    endpointParams: # [array] Endpoint parameters - Optional request parameters to
send to the endpoint.
      - name: # [string] Name - Parameter name
        value: # [string] Value - JavaScript expression to compute the parameter's
value, normally enclosed in backticks (e.g.,Â `${earliest}`). IfÂ a constant, use
single quotes (e.g.,Â 'earliest'). ValuesÃ without delimiters (e.g.,Â earliest) are
evaluated as strings.
    endpointHeaders: # [array] Endpoint headers - Optional request headers to send
to the endpoint.
      - name: # [string] Name - Header Name
        value: # [string] Value - JavaScript expression to compute the header's
value, normally enclosed in backticks (e.g.,Â `${earliest}`). IfÂ a constant, use
single quotes (e.g.,Â 'earliest'). ValuesÃ without delimiters (e.g.,Â earliest) are
evaluated as strings.
    logLevel: # [string] Log level - Collector runtime log Level (verbosity).
    requestTimeout: # [number] Request timeout (seconds) - HTTP request inactivity
timeout, use 0 to disable
    useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS
lookup. When a DNS server returns multiple addresses, this will cause Stream to
cycle through them in the order returned.
    keepAliveTime: # [number] Keep Alive Time (seconds) - How often workers should
check in with the scheduler to keep job subscription alive
    maxMissedKeepAlives: # [number] Worker Timeout (periods) - The number of Keep
Alive Time periods before an inactive worker will have its job subscription revoked.
    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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event
breaking rulesets that will be applied, in order, to the input data stream.
    staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of
time (in milliseconds) the Event Breaker will wait for new data to be sent to a
specific channel, before flushing the data stream out, as-is, to the Pipelines.
    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
that references your credentials

    # ---------------------------------------------------


    # -------------- if authType is textSecret ---------------

    textSecret: # [string] Token (text secret) - Select (or create) a stored text
secret

    # ---------------------------------------------------

    type: # [string] Input Type
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
```

```
id>.
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # ------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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
addresses).
    port: # [number] [required] Port - Port to listen to.
    authTokens: # [array] Auth tokens - Shared secrets to be provided by any client
(Authorization: <token>). If empty, unauthed access is permitted
      - token: # [string] Token - Shared secret to be provided by any client
(Authorization: <token>).
        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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    tls: # [object] TLS settings (server side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      privKeyPath: # [string] Private key path - Path on server containing the
private key to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      certPath: # [string] Certificate path - Path on server containing certificates
to use. PEM format. Can reference $ENV_VARS.
      caPath: # [string] CA certificate path - Path on server containing CA
certificates to use. PEM format. Can reference $ENV_VARS.
      requestCert: # [boolean] Authenticate client (mutual auth) - Whether to
require clients to present their certificates. Used to perform client authentication
using SSL certs.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to accept
from connections.
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept
from connections.

      # ------------------------------------------------------

    maxActiveReq: # [number] Max active requests - Maximum number of active requests
per Worker Process. Use 0 for unlimited.
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
is proxied by a device that supports Proxy Protocol V1 or V2.
    captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add
request headers to events, in the __headers field.
    activityLogSampleRate: # [number] Activity log sample rate - How often request
activity is logged at the `info` level. A value of 1 would log every request, 10
every 10th request, etc.
    requestTimeout: # [number] Request timeout (seconds) - How long to wait for an
incoming request to complete before aborting it. Use 0 to disable.
    splunkHecAPI: # [string] [required] Splunk HEC Endpoint - Absolute path on which
to listen for the Splunk HTTP Event Collector API requests. This input supports the
```

```
/event and /raw endpoints.
    metadata: # [array] Fields - Fields to add to every event. May be overridden by
fields added at the token or request level.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    allowedIndexes: # [array of strings] Allowed Indexes - List values allowed in
HEC event index field, allows wildcards. Leave blank to skip validation.
    splunkHecAcks: # [boolean] Splunk HEC Acks - Whether to enable Splunk HEC
acknowledgements
    breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event
breaking rulesets that will be applied, in order, to the input data stream.
    staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of
time (in milliseconds) the Event Breaker will wait for new data to be sent to a
specific channel, before flushing the data stream out, as-is, to the Pipelines.
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
id>.
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # -------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  azure_blob_input: # [object]
```

```
    type: # [string] Input Type
    queueName: # [string] Queue - The storage account queue name blob notifications
will be read from. Value must be a JavaScript expression (which can evaluate to a
constant value), enclosed in quotes or backticks. Can be evaluated only at init
time. E.g., referencing a Global Variable: `myQueue-${C.vars.myVar}`
    fileFilter: # [string] Filename filter - Regex matching file names to download
and process. Defaults to: .*
    visibilityTimeout: # [number] Visibility timeout (secs) - The duration (in
seconds) that the received messages are hidden from subsequent retrieve requests
after being retrieved by a ReceiveMessage request.
    numReceivers: # [number] Num receivers - The Number of receiver processes to
run, the higher the number the better throughput at the expense of CPU overhead
    maxMessages: # [number] Max messages - The maximum number of messages to return
in a poll request. Azure storage queues never returns more messages than this value
(however, fewer messages might be returned). Valid values: 1 to 32.
    servicePeriodSecs: # [number] Service period (secs) - The duration (in seconds)
which pollers should be validated and restarted if exited
    skipOnError: # [boolean] Skip file on error - Toggle to Yes to skip files that
trigger a processing error. Defaults to No, which enables retries after processing
errors.
    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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event
breaking rulesets that will be applied, in order, to the input data stream.
    staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of
time (in milliseconds) the Event Breaker will wait for new data to be sent to a
specific channel, before flushing the data stream out, as-is, to the Pipelines.
    parquetChunkSizeMB: # [number] Max Parquet chunk size (MB) - Maximum file size
for each Parquet chunk.
    parquetChunkDownloadTimeout: # [number] Parquet chunk download timeout (seconds)
- The maximum time to wait for a Parquet file's chunk to be downloaded. Processing
will end if a required chunk could not be downloaded within the time imposed by this
setting.
    authType: # [string] Authentication method - Enter connection string directly,
or select a stored secret
    connectionString: # [string] Connection string - Enter your Azure Storage
account connection string. If left blank, Stream will fall back to
env.AZURE_STORAGE_CONNECTION_STRING.

    # -------------- if authType is manual ---------------


    # ------------------------------------------------------

    textSecret: # [string] Connection string (text secret) - Select (or create) a
stored text secret

    # -------------- if authType is secret ---------------


    # ------------------------------------------------------

    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.
```

```
    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
id>.
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # ---------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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
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
certificate.
      privKeyPath: # [string] Private key path - Path on server containing the
private key to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      certPath: # [string] Certificate path - Path on server containing certificates
to use. PEM format. Can reference $ENV_VARS.
      caPath: # [string] CA certificate path - Path on server containing CA
certificates to use. PEM format. Can reference $ENV_VARS.
      requestCert: # [boolean] Authenticate client (mutual auth) - Whether to
```

```
require clients to present their certificates. Used to perform client authentication
using SSL certs.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to accept
from connections.
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept
from connections.

      # ------------------------------------------------------

    maxActiveReq: # [number] Max active requests - Maximum number of active requests
per Worker Process. Use 0 for unlimited.
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
is proxied by a device that supports Proxy Protocol V1 or V2.
    captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add
request headers to events, in the __headers field.
    activityLogSampleRate: # [number] Activity log sample rate - How often request
activity is logged at the `info` level. A value of 1 would log every request, 10
every 10th request, etc.
    requestTimeout: # [number] Request timeout (seconds) - How long to wait for an
incoming request to complete before aborting it. Use 0 to disable.
    elasticAPI: # [string] [required] Elasticsearch API endpoint - Absolute path on
which to listen for Elasticsearch API requests. Defaults to /. _bulk will be
appended automatically, e.g., /myPath becomes /myPath/_bulk. Requests can then be
made to either /myPath/_bulk or /myPath/<myIndexName>/_bulk. Other entries are faked
as success.
    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
that references your credentials

    # ------------------------------------------------------


    # -------------- if authType is authTokens ---------------

    authTokens: # [array of strings] Token - Bearer tokens to include in the
authorization header

    # ------------------------------------------------------

    apiVersion: # [string] API Version - The API version to use for communicating
with the server.

    # -------------- if apiVersion is custom ---------------

    customAPIVersion: # [string] Custom API Version - Custom version information to
respond to requests

    # ------------------------------------------------------
```

```
    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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    proxyMode: # [object]
      enabled: # [boolean] Enable Proxy Mode - Enable proxying of non-bulk API
requests to an external Elastic server. Enable this only if you understand the
implications; see docs for more details.

      # -------------- if enabled is true ---------------

      url: # [string] Proxy URL - URL of the Elastic server to proxy non-bulk
requests to, e.g., http://elastic:9200
      removeHeaders: # [array of strings] Remove headers - List of headers to remove
from the request to proxy
      timeoutSec: # [number] Proxy request timeout - Amount of time, in seconds, to
wait for a proxy request to complete before aborting it.
      authType: # [string] Authentication method - Enter credentials directly, or
select a stored secret

      # ------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
```

```
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
id>.
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # ------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  confluent_cloud_input: # [object]
    type: # [string] Input Type
    brokers: # [array of strings] Brokers - List of Confluent Cloud brokers to use,
eg. yourAccount.confluent.cloud:9092
    tls: # [object] TLS settings (client side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to No.
      servername: # [string] Server name (SNI) - Server name for the SNI (Server
Name Indication) TLS extension. It must be a host name, and not an IP address.
      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      caPath: # [string] CA certificate path - Path on client in which to find CA
certificates to verify the server's cert. PEM format. Can reference $ENV_VARS.
      privKeyPath: # [string] Private key path (mutual auth) - Path on client in
which to find the private key to use. PEM format. Can reference $ENV_VARS.
      certPath: # [string] Certificate path (mutual auth) - Path on client in which
to find certificates to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to use when
connecting
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when
connecting

      # ------------------------------------------------------

    topics: # [array of strings] [required] Topic - Topic to subscribe to. Warning:
To optimize performance, Cribl suggests subscribing each Kafka Source to only a
single topic.
    groupId: # [string] Group ID - Specifies the consumer group this instance
belongs to, default is 'Cribl'.
    fromBeginning: # [boolean] From beginning - Whether to start reading from
earliest available data, relevant only during initial subscription.
    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
Confluent Schema Registry, i.e: http://localhost:8081
      tls: # [object] TLS settings (client side)
        disabled: # [boolean] Disabled

        # -------------- if disabled is false ---------------

        rejectUnauthorized: # [boolean] Validate server certs - Reject certs that
are not authorized by a CA in the CA certificate path, or by another trusted CA
```

```
(e.g., the system's CA). Defaults to No.
        servername: # [string] Server name (SNI) - Server name for the SNI (Server
Name Indication) TLS extension. It must be a host name, and not an IP address.
        certificateName: # [string] Certificate name - The name of the predefined
certificate.
        caPath: # [string] CA certificate path - Path on client in which to find CA
certificates to verify the server's cert. PEM format. Can reference $ENV_VARS.
        privKeyPath: # [string] Private key path (mutual auth) - Path on client in
which to find the private key to use. PEM format. Can reference $ENV_VARS.
        certPath: # [string] Certificate path (mutual auth) - Path on client in
which to find certificates to use. PEM format. Can reference $ENV_VARS.
        passphrase: # [string] Passphrase - Passphrase to use to decrypt private
key.
        minVersion: # [string] Minimum TLS version - Minimum TLS version to use when
connecting
        maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when
connecting

        # ------------------------------------------------------


      # ------------------------------------------------------

    connectionTimeout: # [number] Connection timeout (ms) - Maximum time to wait for
a successful connection.
    requestTimeout: # [number] Request timeout (ms) - Maximum time to wait for a
successful request.
    sasl: # [object] Authentication - Authentication parameters to use when
connecting to brokers. Using TLS is highly recommended.
      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
facilities.
      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
rebalance.
      Value must be between the broker's configured group.min.session.timeout.ms and
group.max.session.timeout.ms.
      See details [here]
(https://kafka.apache.org/documentation/#consumerconfigs_session.timeout.ms).
    rebalanceTimeout: # [number] Rebalance timeout (ms) -
      Maximum allowed time for each worker to join the group after a rebalance has
begun.
      If the timeout is exceeded, the coordinator broker will remove the worker from
the group.
      See details [here]
(https://kafka.apache.org/documentation/#connectconfigs_rebalance.timeout.ms).
    heartbeatInterval: # [number] Heartbeat interval (ms) -
      Expected time between heartbeats to the consumer coordinator when using
Kafka's group management facilities.
      Value must be lower than sessionTimeout, and typically should not exceed 1/3
of the sessionTimeout value.
      See details [here]
```

```
(https://kafka.apache.org/documentation/#consumerconfigs_heartbeat.interval.ms).
    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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
id>.
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # ------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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
certificate.
      privKeyPath: # [string] Private key path - Path on server containing the
private key to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      certPath: # [string] Certificate path - Path on server containing certificates
to use. PEM format. Can reference $ENV_VARS.
      caPath: # [string] CA certificate path - Path on server containing CA
certificates to use. PEM format. Can reference $ENV_VARS.
      requestCert: # [boolean] Authenticate client (mutual auth) - Whether to
require clients to present their certificates. Used to perform client authentication
using SSL certs.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to accept
from connections.
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept
from connections.

      # ------------------------------------------------------

    maxActiveReq: # [number] Max active requests - Maximum number of active requests
per Worker Process. Use 0 for unlimited.
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
is proxied by a device that supports Proxy Protocol V1 or V2.
    captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add
request headers to events, in the __headers field.
    activityLogSampleRate: # [number] Activity log sample rate - How often request
activity is logged at the `info` level. A value of 1 would log every request, 10
every 10th request, etc.
    requestTimeout: # [number] Request timeout (seconds) - How long to wait for an
incoming request to complete before aborting it. Use 0 to disable.
    prometheusAPI: # [string] Remote Write API endpoint - Absolute path on which to
listen for Grafana Agent's Remote Write requests. Defaults to /api/prom/push, which
will expand as: http://<yourâupstreamâURL>:<yourâport>/api/prom/push.
    lokiAPI: # [string] 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://<yourâupstreamâURL>:<yourâport>/loki/api/v1/push'.
    keepAliveTimeout: # [number] Keep alive timeout (seconds) - Maximum time to wait
for additional data, after the last response was sent, before closing a socket
connection. This can be very useful when Grafana Agent remote write's request
frequency is high so, reusing connections, would help mitigating the cost of
creating a new connection per request. Note that Grafana Agent's embedded Prometheus
would attempt to keep connections open for up to 5 minutes.
    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
that references your credentials

      # ----------------------------------------------------------


      # -------------- if authType is textSecret ---------------

      textSecret: # [string] Token (text secret) - Select (or create) a stored text
secret

      # ----------------------------------------------------------

    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
that references your credentials

      # ----------------------------------------------------------


      # -------------- if authType is textSecret ---------------

      textSecret: # [string] Token (text secret) - Select (or create) a stored text
secret

      # ----------------------------------------------------------

    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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------
```

```
    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
id>.
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # ------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  loki_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
certificate.
      privKeyPath: # [string] Private key path - Path on server containing the
private key to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      certPath: # [string] Certificate path - Path on server containing certificates
to use. PEM format. Can reference $ENV_VARS.
      caPath: # [string] CA certificate path - Path on server containing CA
certificates to use. PEM format. Can reference $ENV_VARS.
      requestCert: # [boolean] Authenticate client (mutual auth) - Whether to
require clients to present their certificates. Used to perform client authentication
using SSL certs.
```

```
      minVersion: # [string] Minimum TLS version - Minimum TLS version to accept
from connections.
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept
from connections.

      # ------------------------------------------------------

    maxActiveReq: # [number] Max active requests - Maximum number of active requests
per Worker Process. Use 0 for unlimited.
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
is proxied by a device that supports Proxy Protocol V1 or V2.
    captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add
request headers to events, in the __headers field.
    activityLogSampleRate: # [number] Activity log sample rate - How often request
activity is logged at the `info` level. A value of 1 would log every request, 10
every 10th request, etc.
    requestTimeout: # [number] Request timeout (seconds) - How long to wait for an
incoming request to complete before aborting it. Use 0 to disable.
    lokiAPI: # [string] [required] 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://<yourâupstreamâURL>:<yourâport>/loki/api/v1/push'.
    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
that references your credentials

    # -------------------------------------------------------


    # -------------- if authType is textSecret ---------------

    textSecret: # [string] Token (text secret) - Select (or create) a stored text
secret

    # -------------------------------------------------------

    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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
```

```
to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
id>.
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # --------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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
certificate.
      privKeyPath: # [string] Private key path - Path on server containing the
private key to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      certPath: # [string] Certificate path - Path on server containing certificates
to use. PEM format. Can reference $ENV_VARS.
      caPath: # [string] CA certificate path - Path on server containing CA
```

```
certificates to use. PEM format. Can reference $ENV_VARS.
      requestCert: # [boolean] Authenticate client (mutual auth) - Whether to
require clients to present their certificates. Used to perform client authentication
using SSL certs.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to accept
from connections.
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept
from connections.

      # -------------------------------------------------------

    maxActiveReq: # [number] Max active requests - Maximum number of active requests
per Worker Process. Use 0 for unlimited.
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
is proxied by a device that supports Proxy Protocol V1 or V2.
    captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add
request headers to events, in the __headers field.
    activityLogSampleRate: # [number] Activity log sample rate - How often request
activity is logged at the `info` level. A value of 1 would log every request, 10
every 10th request, etc.
    requestTimeout: # [number] Request timeout (seconds) - How long to wait for an
incoming request to complete before aborting it. Use 0 to disable.
    prometheusAPI: # [string] [required] Remote Write API endpoint - Absolute path
on which to listen for Prometheus requests. Defaults to /write, which will expand
as: http://<yourâupstreamâURL>:<yourâport>/write.
    keepAliveTimeout: # [number] Keep alive timeout (seconds) - Maximum time to wait
for additional data, after the last response was sent, before closing a socket
connection. This can be very useful when Prometheus remote write's request frequency
is high so, reusing connections, would help mitigating the cost of creating a new
connection per request. Note that Prometheus would attempt to keep connections open
for up to 5 minutes.
    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
that references your credentials

    # -------------------------------------------------------


    # -------------- if authType is textSecret ---------------

    textSecret: # [string] Token (text secret) - Select (or create) a stored text
secret
```

```
    # ------------------------------------------------------

    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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
id>.
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # ------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  prometheus_input: # [object]
    dimensionList: # [array of strings] Extra Dimensions - Other dimensions to
include in events
    discoveryType: # [string] Discovery Type - Target discovery mechanism. Use
static to manually enter a list of targets.

    # -------------- if discoveryType is static ---------------

    targetList: # [array of strings] Targets - List of Prometheus targets to pull
```

```
metrics from. Values can be in URL or host[:port] format. For example:
http://localhost:9090/metrics, localhost:9090, or localhost. In cases where just
host[:port] is specified, the endpoint will resolve to 'http://host[:port]/metrics'.

    # ------------------------------------------------------


    # -------------- 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
metrics
    scrapePath: # [string] Metrics Path - Path to use when collecting metrics from
discovered targets

    # ------------------------------------------------------


    # -------------- if discoveryType is ec2 ---------------

    usePublicIp: # [boolean] Use Public IP - Use public IP address for discovered
targets. Set to false if the private IP address should be used.
    scrapeProtocol: # [string] Metrics Protocol - Protocol to use when collecting
metrics
    scrapePort: # [number] Metrics Port - The port number in the metrics URL for
discovered targets.
    scrapePath: # [string] Metrics Path - Path to use when collecting metrics from
discovered targets
    searchFilter: # [array] Search Filter - EC2 Instance Search Filter
      - Name: # [string] Filter Name - Search filter attribute name, see:
https://docs.aws.amazon.com/AWSEC2/latest/APIReference/API_DescribeInstances.html
for more information. Attributes can be manually entered if not present in the drop
down list
        Values: # [array of strings] Filter Values - Search Filter Values, if empty
only "running" EC2 instances will be returned
    awsAuthenticationMethod: # [string] Authentication method - AWS authentication
method. Choose Auto to use IAM roles.
    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'
Region-specific endpoint. Otherwise, it must point to EC2-compatible endpoint.
    signatureVersion: # [string] Signature version - Signature version to use for
signing EC2 requests.
    reuseConnections: # [boolean] Reuse connections - Whether to reuse connections
between requests, which can improve performance.
    rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to
reject certificates that cannot be verified against a valid CA (e.g., self-signed
certificates).
    enableAssumeRole: # [boolean] Enable for EC2 - Use Assume Role credentials to
access EC2
    assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the
role to assume
    assumeRoleExternalId: # [string] External ID - External ID to use when assuming
role

    # ------------------------------------------------------

    interval: # [number] Poll Interval - How often in minutes to scrape targets for
metrics, 60 must be evenly divisible by the value or save will fail.
```

```
    logLevel: # [string] [required] Log Level - Collector runtime Log Level
    keepAliveTime: # [number] Keep Alive Time (seconds) - How often workers should
check in with the scheduler to keep job subscription alive
    maxMissedKeepAlives: # [number] Worker Timeout (periods) - The number of Keep
Alive Time periods before an inactive worker will have its job subscription revoked.
    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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    authType: # [string] Authentication method - Enter credentials directly, or
select a stored secret
    username: # [string] Username - Username for Prometheus Basic authentication
    password: # [string] Password - Password for Prometheus Basic authentication

    # -------------- if authType is manual ---------------


    # -----------------------------------------------------

    credentialsSecret: # [string] Credentials secret - Select (or create) a secret
that references your credentials

    # -------------- 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
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
```

```
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
id>.
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # ------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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
disable
    keepAliveTime: # [number] Keep Alive Time (seconds) - How often workers should
check in with the scheduler to keep job subscription alive
    maxMissedKeepAlives: # [number] Worker Timeout (periods) - The number of Keep
Alive Time periods before an inactive worker will have its job subscription revoked.
    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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    planType: # [string] [required] Subscription Plan - Office 365 subscription plan
for your organization, typically Enterprise
    publisherIdentifier: # [string] Publisher Identifier - Optional Publisher
Identifier to use in API requests, defaults to tenant id if not defined. For more
information see [here](https://docs.microsoft.com/en-us/office/office-365-
management-api/office-365-management-activity-api-reference#start-a-subscription)
    contentConfig: # [array] Content Types - Enable Office 365 Management Activity
API content types and polling intervals. Polling intervals are used to set up search
date range and cron schedule, e.g.: */${interval} * * * *. Because of this,
intervals entered must be evenly divisible by 60 to give a predictable schedule.
      - contentType: # [string] Content Type - Office 365 Management Activity API
Content Type
        description: # [string] Interval Description - If interval type is minutes
the value entered must evenly divisible by 60 or save will fail
        interval: # [number] Interval
        logLevel: # [string] Log Level - Collector runtime Log Level
        enabled: # [boolean] Enabled
    authType: # [string] Authentication method - Enter client secret directly, or
select a stored secret
    clientSecret: # [string] Client secret - Office 365 Azure client secret

    # -------------- if authType is manual ---------------


    # ------------------------------------------------------

    textSecret: # [string] Client secret (text secret) - Select (or create) a stored
text secret

    # -------------- if authType is secret ---------------


    # ------------------------------------------------------
```

```
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
id>.
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # --------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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
disable
    keepAliveTime: # [number] Keep Alive Time (seconds) - How often workers should
check in with the scheduler to keep job subscription alive
    maxMissedKeepAlives: # [number] Worker Timeout (periods) - The number of Keep
Alive Time periods before an inactive worker will have its job subscription revoked.
    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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
```

```
    contentConfig: # [array] Content Types - Enable Office 365 Service Communication
API content types and polling intervals. Polling intervals are used to set up search
date range and cron schedule, e.g.: */${interval} * * * *. Because of this,
intervals entered for current and historical status must be evenly divisible by 60
to give a predictable schedule.
      - contentType: # [string] Content Type - Office 365 Services API Content Type
        description: # [string] Interval Description - If interval type is minutes
the value entered must evenly divisible by 60 or save will fail
        interval: # [number] Interval
        logLevel: # [string] Log Level - Collector runtime Log Level
        enabled: # [boolean] Enabled
    authType: # [string] Authentication method - Enter client secret directly, or
select a stored secret
    clientSecret: # [string] Client secret - Office 365 Azure client secret

    # -------------- if authType is manual ---------------


    # ------------------------------------------------------

    textSecret: # [string] Client secret (text secret) - Select (or create) a stored
text secret

    # -------------- if authType is secret ---------------


    # ------------------------------------------------------

    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
```

```
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
id>.
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # ------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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
the report. Must divide evenly into 60 minutes to create a predictable schedule, or
Save will fail.
    startDate: # [string] Date range start - Backward offset for the search range's
head. (E.g.: -3h@h) Message Trace data is delayed; this parameter (with Date range
end) compensates for delay and gaps.
    endDate: # [string] Date range end - Backward offset for the search range's
tail. (E.g.: -2h@h) Message Trace data is delayed; this parameter (with Date range
start) compensates for delay and gaps.
    logLevel: # [string] Log level - Log Level (verbosity) for collection runtime
behavior.
    timeout: # [number] Timeout (secs) - HTTP request inactivity timeout. Maximum is
2400 (40 minutes); enter 0 to wait indefinitely.
    disableTimeFilter: # [boolean] Disable time filter - Disables time filtering of
events when a date range is specified.
    authType: # [string] Authentication method - Select authentication method.

    # -------------- if authType is manual ---------------

    username: # [string] Username - Username to run Message Trace API call.
    password: # [string] Password - Password to run Message Trace API call.

    # ------------------------------------------------------


    # -------------- if authType is secret ---------------

    credentialsSecret: # [string] Credentials secret - Select (or create) a secret
that references your credentials.

    # ------------------------------------------------------


    # -------------- if authType is oauth ---------------

    clientSecret: # [string] Client secret - client_secret to pass in the OAuth
request parameter.
    tenantId: # [string] Tenant identifier - Directory ID (tenant identifier) in
Azure Active Directory.
    clientId: # [string] Client ID - client_id to pass in the OAuth request
parameter.
    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 your client_secret to pass in the OAuth request parameter.
    tenantId: # [string] Tenant identifier - Directory ID (tenant identifier) in
Azure Active Directory.
    clientId: # [string] Client ID - client_id to pass in the OAuth request
parameter.
    resource: # [string] Resource - Resource to pass in the OAuth request parameter.

    # ----------------------------------------------------------

    keepAliveTime: # [number] Keep Alive Time (seconds) - How often workers should
check in with the scheduler to keep job subscription alive
    maxMissedKeepAlives: # [number] Worker Timeout (periods) - The number of Keep
Alive Time periods before an inactive worker will have its job subscription revoked.
    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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    type: # [string] Input Type
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
id>.
```

```
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # ------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  eventhub_input: # [object]
    type: # [string] Input Type
    brokers: # [array of strings] Brokers - List of Event Hubs Kafka brokers to
connect to, e.g., yourdomain.servicebus.windows.net:9093. The hostname can be found
in the host portion of the primary or secondary connection string in Shared Access
Policies.
    topics: # [array of strings] [required] Event Hub name - The name of the Event
Hub (a.k.a. Kafka topic) to subscribe to. Warning: To optimize performance, Cribl
suggests subscribing each Event Hubs Source to only a single topic.
    groupId: # [string] Group ID - Specifies the consumer group this instance
belongs to, default is 'Cribl'.
    fromBeginning: # [boolean] From beginning - Whether to start reading from
earliest available data, relevant only during initial subscription.
    connectionTimeout: # [number] Connection timeout (ms) - Maximum time to wait for
a successful connection.
    requestTimeout: # [number] Request timeout (ms) - Maximum time to wait for a
successful request.
    sasl: # [object] Authentication - Authentication parameters to use when
connecting to brokers. Using TLS is highly recommended.
      disabled: # [boolean] Disabled - Enable authentication.

      # -------------- if disabled is false ---------------

      mechanism: # [string] SASL mechanism - SASL authentication mechanism to use.
PLAIN is the only mechanism currently supported for Event Hubs Kafka brokers.
      username: # [string] Username - The username for authentication. For Event
Hubs, this should always be $ConnectionString.
      authType: # [string] Authentication method - Enter password directly, or
select a stored secret

      # ------------------------------------------------------

    tls: # [object] TLS settings (client side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      rejectUnauthorized: # [boolean] Validate server certs - For Event Hubs, this
should always be false.

      # ------------------------------------------------------

    sessionTimeout: # [number] Session timeout (ms) -
      Timeout (a.k.a session.timeout.ms in Kafka domain) used to detect client
failures when using Kafka's group management facilities.
      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 rebalance.
      Value must be lower than rebalanceTimeout.
      See details [here](https://github.com/Azure/azure-event-hubs-for-
kafka/blob/master/CONFIGURATION.md).
    rebalanceTimeout: # [number] Rebalance timeout (ms) -
      Maximum allowed time (a.k.a rebalance.timeout.ms in Kafka domain) for each
worker to join the group after a rebalance has begun.
```

```
      If the timeout is exceeded, the coordinator broker will remove the worker from
the group.
      See details [here](https://github.com/Azure/azure-event-hubs-for-
kafka/blob/master/CONFIGURATION.md).
    heartbeatInterval: # [number] Heartbeat interval (ms) -
      Expected time (a.k.a heartbeat.interval.ms in Kafka domain) between heartbeats
to the consumer coordinator when using Kafka's group management facilities.
      Value must be lower than sessionTimeout, and typically should not exceed 1/3
of the sessionTimeout value.
      See details [here](https://github.com/Azure/azure-event-hubs-for-
kafka/blob/master/CONFIGURATION.md).
    minimizeDuplicates: # [boolean] Minimize duplicates - Enable feature to minimize
duplicate events by only starting one consumer for each topic partition.
    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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
id>.
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # ------------------------------------------------------
```

```
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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
that the command fails.
    scheduleType: # [string] Schedule type - Select a schedule type; either an
interval (in seconds) or a cron-style schedule.

    # -------------- 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
breaking rulesets that will be applied, in order, to the input data stream.
    staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of
time (in milliseconds) the Event Breaker will wait for new data to be sent to a
specific channel, before flushing the data stream out, as-is, to the Pipelines.
    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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    type: # [string] Input Type
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
```

```
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
id>.
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # ------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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
any client (Authorization: <token>). If empty, unauthed access is permitted.
    tls: # [object] TLS settings (server side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      privKeyPath: # [string] Private key path - Path on server containing the
private key to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      certPath: # [string] Certificate path - Path on server containing certificates
to use. PEM format. Can reference $ENV_VARS.
      caPath: # [string] CA certificate path - Path on server containing CA
certificates to use. PEM format. Can reference $ENV_VARS.
      requestCert: # [boolean] Authenticate client (mutual auth) - Whether to
require clients to present their certificates. Used to perform client authentication
using SSL certs.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to accept
from connections.
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept
from connections.

      # ------------------------------------------------------

    maxActiveReq: # [number] Max active requests - Maximum number of active requests
per Worker Process. Use 0 for unlimited.
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
is proxied by a device that supports Proxy Protocol V1 or V2.
    captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add
request headers to events, in the __headers field.
    activityLogSampleRate: # [number] Activity log sample rate - How often request
activity is logged at the `info` level. A value of 1 would log every request, 10
every 10th request, etc.
    requestTimeout: # [number] Request timeout (seconds) - How long to wait for an
incoming request to complete before aborting it. Use 0 to disable.
```

```
    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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
id>.
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # --------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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
to use when receiving events.
    createTopic: # [boolean] Create topic - If enabled, create topic if it does not
exist
    createSubscription: # [boolean] Create subscription - If enabled, create
subscription if it does not exist

    # -------------- if createSubscription is true ---------------
```

```
    orderedDelivery: # [boolean] Ordered delivery - If enabled, receive events in
the order they were added to the queue. For this to work correctly, the process
sending events must have ordering enabled.

    # ------------------------------------------------------

    region: # [string] Region - Region to retrieve messages from. Select 'default'
to allow Google to auto-select the nearest region. When using ordered delivery, the
selected region must be allowed by message storage policy.
    googleAuthMethod: # [string] Authentication Method - Google authentication
method. Choose Auto to use environment variables PUBSUB_PROJECT and
PUBSUB_CREDENTIALS.

    # -------------- if googleAuthMethod is manual ---------------

    serviceAccountCredentials: # [string] Service account credentials - Contents of
service account credentials (JSON keys) file downloaded from Google Cloud. To upload
a file, click the upload button at this field's upper right. As an alternative, you
can use environment variables (see [here](https://googleapis.dev/ruby/google-cloud-
pubsub/latest/file.AUTHENTICATION.html)).

    # ------------------------------------------------------


    # -------------- if googleAuthMethod is secret ---------------

    secret: # [string] Service account credentials (text secret) - Select (or
create) a stored text secret

    # ------------------------------------------------------

    maxBacklog: # [number] Max backlog - If Destination exerts backpressure, this
setting limits how many inbound events Stream will queue for processing before it
stops retrieving events.
    requestTimeout: # [number] Request timeout (ms) - Pull request timeout, in
milliseconds.
    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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------
```

```
    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
id>.
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # ------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
```

```
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
id>.
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # ------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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
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
certificate.
      privKeyPath: # [string] Private key path - Path on server containing the
private key to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      certPath: # [string] Certificate path - Path on server containing certificates
to use. PEM format. Can reference $ENV_VARS.
      caPath: # [string] CA certificate path - Path on server containing CA
certificates to use. PEM format. Can reference $ENV_VARS.
      requestCert: # [boolean] Authenticate client (mutual auth) - Whether to
require clients to present their certificates. Used to perform client authentication
using SSL certs.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to accept
from connections.
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept
from connections.

      # ------------------------------------------------------

    maxActiveCxn: # [number] Max Active Connections - Maximum number of active
connections allowed per Worker Process, use 0 for unlimited
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
is proxied by a device that supports Proxy Protocol V1 or V2.
    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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------
```

```
    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
id>.
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # ------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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
any client (Authorization: <token>). If empty, unauthed access is permitted.
    tls: # [object] TLS settings (server side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      privKeyPath: # [string] Private key path - Path on server containing the
private key to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      certPath: # [string] Certificate path - Path on server containing certificates
to use. PEM format. Can reference $ENV_VARS.
      caPath: # [string] CA certificate path - Path on server containing CA
certificates to use. PEM format. Can reference $ENV_VARS.
```

```
      requestCert: # [boolean] Authenticate client (mutual auth) - Whether to
require clients to present their certificates. Used to perform client authentication
using SSL certs.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to accept
from connections.
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept
from connections.

      # ----------------------------------------------------------

    maxActiveReq: # [number] Max active requests - Maximum number of active requests
per Worker Process. Use 0 for unlimited.
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
is proxied by a device that supports Proxy Protocol V1 or V2.
    captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add
request headers to events, in the __headers field.
    activityLogSampleRate: # [number] Activity log sample rate - How often request
activity is logged at the `info` level. A value of 1 would log every request, 10
every 10th request, etc.
    requestTimeout: # [number] Request timeout (seconds) - How long to wait for an
incoming request to complete before aborting it. Use 0 to disable.
    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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
```

```
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
id>.
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # ------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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
certificate.
      privKeyPath: # [string] Private key path - Path on server containing the
private key to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      certPath: # [string] Certificate path - Path on server containing certificates
to use. PEM format. Can reference $ENV_VARS.
      caPath: # [string] CA certificate path - Path on server containing CA
certificates to use. PEM format. Can reference $ENV_VARS.
      requestCert: # [boolean] Authenticate client (mutual auth) - Whether to
require clients to present their certificates. Used to perform client authentication
using SSL certs.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to accept
from connections.
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept
from connections.

      # ------------------------------------------------------

    ipWhitelistRegex: # [string] IP Allowlist Regex - Regex matching IP addresses
that are allowed to establish a connection.
    maxActiveCxn: # [number] Max Active Connections - Maximum number of active
connections allowed per Worker Process, use 0 for unlimited
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
is proxied by a device that supports Proxy Protocol V1 or V2.
    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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    authType: # [string] Authentication method - Enter a token directly, or provide
a secret referencing a token
    authToken: # [string] Auth token - Shared secret to be provided by any client
(in authToken header field). If empty, unauthed access is permitted.

    # -------------- if authType is manual ---------------


    # ------------------------------------------------------

    textSecret: # [string] Auth token (text secret) - Select (or create) a stored
```

```
text secret

    # -------------- if authType is secret ---------------


    # ------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
id>.
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # ------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  system_metrics_input: # [object]
    interval: # [number] Polling interval - Time, in seconds, between consecutive
metric collections.
    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 ---------------

          processes: # [boolean] Process metrics - Generate metrics for the numbers
of processes in various states

          # ------------------------------------------------------

        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
counters

          # ------------------------------------------------------

        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
states

          # ------------------------------------------------------

        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
include/exclude. E.g.: eth0, !lo, etc. All interfaces are included if this list is
empty.
          perInterface: # [boolean] Per interface metrics - Generate separate
metrics for each interface
          detail: # [boolean] Detailed metrics - Generate full network metrics

          # ------------------------------------------------------

        disk: # [object]
          mode: # [string]  - Select the level of details for disk metrics

          # -------------- if mode is custom ---------------

          devices: # [array of strings] Device filter - Block devices to
include/exclude. E.g.: sda*, !loop*, etc. Wildcards and ! (not) operators are
supported. All devices are included if this list is empty.
          mountpoints: # [array of strings] Mountpoint filter - Filesystem
mountpoints to include/exclude. E.g.: /, /home, !/proc*, !/tmp, etc. Wildcards and !
(not) operators are supported. All mountpoints are included if this list is empty.
          fstypes: # [array of strings] Filesystem type filter - Filesystem types to
include/exclude. E.g.: ext4, !*tmpfs, !squashfs, etc. Wildcards and ! (not)
operators are supported. All types are included if this list is empty.
```

```
          perDevice: # [boolean] Per device metrics - Generate separate metrics for
each device
          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
UNIX-domain socket
      dockerTimeout: # [number] Docker timeout - Timeout, in seconds, for the Docker
API

      # ------------------------------------------------------


      # -------------- if mode is all ---------------

      dockerSocket: # [array of strings] Docker socket - Full paths for Docker's
UNIX-domain socket
      dockerTimeout: # [number] Docker timeout - Timeout, in seconds, for the Docker
API

      # ------------------------------------------------------


      # -------------- if mode is custom ---------------

      filters: # [array] Container Filters - Containers matching any of these will
be included. All are included if this is empty.
        - expr: # [string] Expression
      allContainers: # [boolean] All containers - Include stopped and paused
containers
      perDevice: # [boolean] Per device metrics - Generate separate metrics for each
device
      detail: # [boolean] Detailed metrics - Generate full container metrics
      dockerSocket: # [array of strings] Docker socket - Full paths for Docker's
UNIX-domain socket
      dockerTimeout: # [number] Docker timeout - Timeout, in seconds, for the Docker
API

      # ------------------------------------------------------

    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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    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
```

```
consumed (e.g., 420MB or 4GB). Once reached, older data will be deleted.
      maxDataTime: # [string] Max data age - Maximum amount of time to retain data
(e.g., 2h or 4d). Once reached, older data will be deleted.
      compress: # [string] Compression - Select data compression format. Optional.
      destPath: # [string] Path location - Path to use to write metrics. Defaults to
$CRIBL_HOME/state/<id>`

      # ------------------------------------------------------

    type: # [string] Input Type
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
id>.
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # -------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  system_state_input: # [object]
    interval: # [number] Polling interval - Time, in seconds, between consecutive
state collections.
    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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    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
consumed (e.g., 420MB or 4GB). Once reached, older data will be deleted.
      maxDataTime: # [string] Max data age - Maximum amount of time to retain data
(e.g., 2h or 4d). Once reached, older data will be deleted.
      compress: # [string] Compression - Select data compression format. Optional.
      destPath: # [string] Path location - Path to use to write metrics. Defaults to
$CRIBL_HOME/state/<id>`

      # ------------------------------------------------------

    type: # [string] Input Type
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
id>.
```

```
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # ------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  windows_metrics_input: # [object]
    interval: # [number] Polling interval - Time, in seconds, between consecutive
metric collections.
    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
information

          # ------------------------------------------------------

        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
counters

          # ------------------------------------------------------

        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
states

          # ------------------------------------------------------

        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
include/exclude. All interfaces are included if this list is empty.
          perInterface: # [boolean] Per interface metrics - Generate separate
metrics for each interface
          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/exclude. E.g.: C:, !E:, etc. Wildcards and ! (not) operators are supported.
All volumes are included if this list is empty.
          perVolume: # [boolean] Per volume metrics - Generate separate metrics for
each volume

          # -----------------------------------------------------


      # -------------------------------------------------------

    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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    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
consumed (e.g., 420MB or 4GB). Once reached, older data will be deleted.
      maxDataTime: # [string] Max data age - Maximum amount of time to retain data
(e.g., 2h or 4d). Once reached, older data will be deleted.
      compress: # [string] Compression - Select data compression format. Optional.
      destPath: # [string] Path location - Path to use to write metrics. Defaults to
$CRIBL_HOME/state/<id>`

      # -------------------------------------------------------

    type: # [string] Input Type
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
```

```
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
id>.
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # ------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  crowdstrike_input: # [object]
    type: # [string] Input Type
    queueName: # [string] Queue - The name, URL, or ARN of the SQS queue to read
notifications from. When a non-AWS URL is specified, format must be:
'{url}/myQueueName'. E.g., 'https://host:port/myQueueName'. Value must be a
JavaScript expression (which can evaluate to a constant value), enclosed in quotes
or backticks. Can be evaluated only at init time. E.g., referencing a Global
Variable: `https://host:port/myQueue-${C.vars.myVar}`.
    fileFilter: # [string] Filename filter - Regex matching file names to download
and process. Defaults to: .*
    awsAccountId: # [string] AWS Account ID - SQS queue owner's AWS account ID.
Leave empty if SQS queue is in same AWS account.
    awsAuthenticationMethod: # [string] Authentication method - AWS authentication
method. Choose Auto to use IAM roles.

    # -------------- 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
references your access key and secret key.

    # ------------------------------------------------------

    awsSecretKey: # [string] Secret key - Secret key
    region: # [string] Region - AWS Region where the S3 bucket and SQS queue are
located. Required, unless the Queue entry is a URL or ARN that includes a Region.
    endpoint: # [string] Endpoint - S3 service endpoint. If empty, defaults to AWS'
Region-specific endpoint. Otherwise, it must point to S3-compatible endpoint.
    signatureVersion: # [string] Signature version - Signature version to use for
signing S3 requests.
    reuseConnections: # [boolean] Reuse connections - Whether to reuse connections
```

```
between requests, which can improve performance.
    rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to
reject certificates that cannot be verified against a valid CA (e.g., self-signed
certificates).
    breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event
breaking rulesets that will be applied, in order, to the input data stream.
    staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of
time (in milliseconds) the Event Breaker will wait for new data to be sent to a
specific channel, before flushing the data stream out, as-is, to the Pipelines.
    maxMessages: # [number] Max Messages - The maximum number of messages SQS should
return in a poll request. Amazon SQS never returns more messages than this value
(however, fewer messages might be returned). Valid values: 1 to 10.
    visibilityTimeout: # [number] Visibility timeout seconds - The duration (in
seconds) that received messages are hidden from subsequent retrieve requests, after
being retrieved by a ReceiveMessage request. This value also automatically extends
this duration, to prevent it from expiring before processing completes.
    numReceivers: # [number] Num receivers - The Number of receiver processes to
run, the higher the number the better throughput at the expense of CPU overhead
    socketTimeout: # [number] Socket timeout - Socket inactivity timeout (in
seconds). Increase this value if timeouts occur due to backpressure.
    skipOnError: # [boolean] Skip file on error - Toggle to Yes to skip files that
trigger a processing error. Defaults to No, which enables retries after processing
errors.
    enableAssumeRole: # [boolean] Enable for S3 - Use Assume Role credentials to
access S3
    assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the
role to assume
    assumeRoleExternalId: # [string] External ID - External ID to use when assuming
role
    enableSQSAssumeRole: # [boolean] Enable for SQS - Use Assume Role credentials
when accessing SQS.
    preprocess: # [object]
      disabled: # [boolean] Disabled - Enable Custom Command
      command: # [string] Command - Command to feed the data through (via stdin) and
process its output (stdout)
      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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    pollTimeout: # [number] Poll timeout (secs) - The amount of time to wait for
events before trying polling again. The lower the number the higher the AWS bill.
The higher the number the longer it will take for the source to react to
configuration changes and system restarts.
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
```

```
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
id>.
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # ------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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
certificate.
      privKeyPath: # [string] Private key path - Path on server containing the
private key to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      certPath: # [string] Certificate path - Path on server containing certificates
to use. PEM format. Can reference $ENV_VARS.
      caPath: # [string] CA certificate path - Path on server containing CA
certificates to use. PEM format. Can reference $ENV_VARS.
      requestCert: # [boolean] Authenticate client (mutual auth) - Whether to
require clients to present their certificates. Used to perform client authentication
using SSL certs.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to accept
from connections.
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept
from connections.

      # ------------------------------------------------------

    maxActiveReq: # [number] Max active requests - Maximum number of active requests
```

```
per Worker Process. Use 0 for unlimited.
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
is proxied by a device that supports Proxy Protocol V1 or V2.
    captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add
request headers to events, in the __headers field.
    activityLogSampleRate: # [number] Activity log sample rate - How often request
activity is logged at the `info` level. A value of 1 would log every request, 10
every 10th request, etc.
    requestTimeout: # [number] Request timeout (seconds) - How long to wait for an
incoming request to complete before aborting it. Use 0 to disable.
    extractMetrics: # [boolean] 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).
    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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    proxyMode: # [object]
      enabled: # [boolean] 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.
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
id>.
```

```
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # ------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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
events to generate per second per worker node. Defaults to 10.
    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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    type: # [string] Input Type
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
id>.
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # ------------------------------------------------------
```

```
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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
addresses).
    port: # [number] [required] Port - Port to listen to.
    authTokens: # [array of strings] Auth tokens - Shared secrets to be provided by
any client (Authorization: <token>). If empty, unauthed access is permitted.
    tls: # [object] TLS settings (server side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      privKeyPath: # [string] Private key path - Path on server containing the
private key to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      certPath: # [string] Certificate path - Path on server containing certificates
to use. PEM format. Can reference $ENV_VARS.
      caPath: # [string] CA certificate path - Path on server containing CA
certificates to use. PEM format. Can reference $ENV_VARS.
      requestCert: # [boolean] Authenticate client (mutual auth) - Whether to
require clients to present their certificates. Used to perform client authentication
using SSL certs.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to accept
from connections.
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept
from connections.

      # ----------------------------------------------------------

    maxActiveReq: # [number] Max active requests - Maximum number of active requests
per Worker Process. Use 0 for unlimited.
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
is proxied by a device that supports Proxy Protocol V1 or V2.
    captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add
request headers to events, in the __headers field.
    activityLogSampleRate: # [number] Activity log sample rate - How often request
activity is logged at the `info` level. A value of 1 would log every request, 10
every 10th request, etc.
    requestTimeout: # [number] Request timeout (seconds) - How long to wait for an
incoming request to complete before aborting it. Use 0 to disable.
    breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event
breaking rulesets that will be applied, in order, to the input data stream.
    staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of
time (in milliseconds) the Event Breaker will wait for new data to be sent to a
specific channel, before flushing the data stream out, as-is, to the Pipelines.
    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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    allowedPaths: # [array of strings] Allowed URI paths - List of URI paths
accepted by this input, wildcards are supported, e.g /api/v*/hook. Defaults to allow
all.
    allowedMethods: # [array of strings] Allowed HTTP methods - List of HTTP methods
accepted by this input, wildcards are supported, e.g. P*, GET. Defaults to allow
all.
```

```
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
id>.
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # -------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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
consecutive service calls
    shardExpr: # [string] Shard selection expression - A JS expression to be called
with each shardId for the stream, if the expression evalutates to a truthy value the
shard will be processed.
    shardIteratorType: # [string] Shard iterator start - Location at which to start
reading a shard for the first time.
    payloadFormat: # [string] Record data format - Format of data inside the Kinesis
Stream records. Gzip compression is automatically detected.
    awsAuthenticationMethod: # [string] Authentication method - AWS authentication
method. Choose Auto to use IAM roles.

    # -------------- 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
references your access key and secret key.

    # ------------------------------------------------------

    awsSecretKey: # [string] Secret key - Secret key
    region: # [string] [required] Region - Region where the Kinesis stream is
located
    endpoint: # [string] Endpoint - Kinesis stream service endpoint. If empty,
defaults to AWS' Region-specific endpoint. Otherwise, it must point to Kinesis
stream-compatible endpoint.
    signatureVersion: # [string] Signature version - Signature version to use for
signing Kinesis stream requests.
    reuseConnections: # [boolean] Reuse connections - Whether to reuse connections
between requests, which can improve performance.
    rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to
reject certificates that cannot be verified against a valid CA (e.g., self-signed
certificates).
    enableAssumeRole: # [boolean] Enable for Kinesis stream - Use Assume Role
credentials to access Kinesis stream
    assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the
role to assume
    assumeRoleExternalId: # [string] External ID - External ID to use when assuming
role
    verifyKPLCheckSums: # [boolean] Verify KPL checksums - Verify Kinesis Producer
Library (KPL) event checksums
    avoidDuplicates: # [boolean] Avoid duplicate records - Yes means: when resuming
streaming from a stored state, Stream will read the next available record, rather
than rereading the last-read record. Enabling this can cause data loss after a
Worker Node's unexpected shutdown or restart.
    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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue
```

```
    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
id>.
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # ----------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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
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
certificate.
      privKeyPath: # [string] Private key path - Path on server containing the
private key to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      certPath: # [string] Certificate path - Path on server containing certificates
to use. PEM format. Can reference $ENV_VARS.
      caPath: # [string] CA certificate path - Path on server containing CA
certificates to use. PEM format. Can reference $ENV_VARS.
      requestCert: # [boolean] Authenticate client (mutual auth) - Whether to
require clients to present their certificates. Used to perform client authentication
using SSL certs.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to accept
from connections.
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept
from connections.

      # ----------------------------------------------------

    ipWhitelistRegex: # [string] IP Allowlist Regex - Regex matching IP addresses
that are allowed to establish a connection.
    maxActiveCxn: # [number] Max Active Connections - Maximum number of active
```

```
connections allowed per Worker Process, use 0 for unlimited
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
is proxied by a device that supports Proxy Protocol V1 or V2.
    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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    authType: # [string] Authentication method - Enter a token directly, or provide
a secret referencing a token
    authToken: # [string] Auth token - Shared secret to be provided by any client
(in authToken header field). If empty, unauthed access is permitted.

    # -------------- if authType is manual ---------------


    # -----------------------------------------------------

    textSecret: # [string] Auth token (text secret) - Select (or create) a stored
text secret

    # -------------- if authType is secret ---------------


    # -----------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
```

```
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
id>.
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # ------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  criblmetrics_input: # [object]
    type: # [string] Input Type
    prefix: # [string] Metric Name Prefix - A prefix that is applied to the metrics
provided by Cribl Stream
    fullFidelity: # [boolean] Full Fidelity - Include granular metrics.  Disabling
this will drop the following metrics events: `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)`.
    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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
```

```
id>.
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # ------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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
default '0.0.0.0'. For IPv6, enter '::' (all addresses) or specify an IP address.
    udpPort: # [number] UDP Port - Enter UDP port number to listen on. Not required
if listening on TCP.
    tcpPort: # [number] TCP Port - Enter TCP port number to listen on. Not required
if listening on UDP.
    maxBufferSize: # [number] Max Buffer Size (events) - Maximum number of events to
buffer when downstream is blocking. Only applies to UDP.
    ipWhitelistRegex: # [string] IP Allowlist Regex - Regex matching IP addresses
that are allowed to send data
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
is proxied by a device that supports Proxy Protocol V1 or V2
    tls: # [object] TLS settings (server side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      privKeyPath: # [string] Private key path - Path on server containing the
private key to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      certPath: # [string] Certificate path - Path on server containing certificates
to use. PEM format. Can reference $ENV_VARS.
      caPath: # [string] CA certificate path - Path on server containing CA
certificates to use. PEM format. Can reference $ENV_VARS.
      requestCert: # [boolean] Authenticate client (mutual auth) - Whether to
require clients to present their certificates. Used to perform client authentication
using SSL certs.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to accept
from connections.
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept
from connections.

      # ------------------------------------------------------

    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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
```

```
      - 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
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
id>.
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # ------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  s3_input: # [object]
    type: # [string] Input Type
    queueName: # [string] Queue - The name, URL, or ARN of the SQS queue to read
notifications from. When a non-AWS URL is specified, format must be:
'{url}/myQueueName'. E.g., 'https://host:port/myQueueName'. Value must be a
JavaScript expression (which can evaluate to a constant value), enclosed in quotes
or backticks. Can be evaluated only at init time. E.g., referencing a Global
Variable: `https://host:port/myQueue-${C.vars.myVar}`.
    fileFilter: # [string] Filename filter - Regex matching file names to download
and process. Defaults to: .*
    awsAccountId: # [string] AWS Account ID - SQS queue owner's AWS account ID.
Leave empty if SQS queue is in same AWS account.
    awsAuthenticationMethod: # [string] Authentication method - AWS authentication
method. Choose Auto to use IAM roles.

    # -------------- 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
```

```
references your access key and secret key.

    # ------------------------------------------------------

    awsSecretKey: # [string] Secret key - Secret key
    region: # [string] Region - AWS Region where the S3 bucket and SQS queue are
located. Required, unless the Queue entry is a URL or ARN that includes a Region.
    endpoint: # [string] Endpoint - S3 service endpoint. If empty, defaults to AWS'
Region-specific endpoint. Otherwise, it must point to S3-compatible endpoint.
    signatureVersion: # [string] Signature version - Signature version to use for
signing S3 requests.
    reuseConnections: # [boolean] Reuse connections - Whether to reuse connections
between requests, which can improve performance.
    rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to
reject certificates that cannot be verified against a valid CA (e.g., self-signed
certificates).
    breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event
breaking rulesets that will be applied, in order, to the input data stream.
    staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of
time (in milliseconds) the Event Breaker will wait for new data to be sent to a
specific channel, before flushing the data stream out, as-is, to the Pipelines.
    maxMessages: # [number] Max Messages - The maximum number of messages SQS should
return in a poll request. Amazon SQS never returns more messages than this value
(however, fewer messages might be returned). Valid values: 1 to 10.
    visibilityTimeout: # [number] Visibility timeout seconds - The duration (in
seconds) that received messages are hidden from subsequent retrieve requests, after
being retrieved by a ReceiveMessage request. This value also automatically extends
this duration, to prevent it from expiring before processing completes.
    numReceivers: # [number] Num receivers - The Number of receiver processes to
run, the higher the number the better throughput at the expense of CPU overhead
    socketTimeout: # [number] Socket timeout - Socket inactivity timeout (in
seconds). Increase this value if timeouts occur due to backpressure.
    skipOnError: # [boolean] Skip file on error - Toggle to Yes to skip files that
trigger a processing error. Defaults to No, which enables retries after processing
errors.
    enableAssumeRole: # [boolean] Enable for S3 - Use Assume Role credentials to
access S3
    assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the
role to assume
    assumeRoleExternalId: # [string] External ID - External ID to use when assuming
role
    enableSQSAssumeRole: # [boolean] Enable for SQS - Use Assume Role credentials
when accessing SQS.
    preprocess: # [object]
      disabled: # [boolean] Disabled - Enable Custom Command
      command: # [string] Command - Command to feed the data through (via stdin) and
process its output (stdout)
      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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    parquetChunkSizeMB: # [number] Max Parquet chunk size (MB) - Maximum file size
for each Parquet chunk.
    parquetChunkDownloadTimeout: # [number] Parquet chunk download timeout (seconds)
- The maximum time to wait for a Parquet file's chunk to be downloaded. Processing
will end if a required chunk could not be downloaded within the time imposed by this
setting.
    pollTimeout: # [number] Poll timeout (secs) - The amount of time to wait for
events before trying polling again. The lower the number the higher the AWS bill.
```

```
The higher the number the longer it will take for the source to react to
configuration changes and system restarts.
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
id>.
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # --------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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
default '0.0.0.0'. For IPv6, enter '::' (all addresses) or specify an IP address.
    port: # [number] [required] UDP Port - UDP port to receive SNMP traps on.
Defaults to 162.
    maxBufferSize: # [number] Max Buffer Size (events) - Maximum number of events to
buffer when downstream is blocking.
    ipWhitelistRegex: # [string] IP Allowlist Regex - Regex matching IP addresses
that are allowed to send data
    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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
id>.
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # --------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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
certificate.
      privKeyPath: # [string] Private key path - Path on server containing the
```

```
private key to use. PEM format. Can reference $ENV_VARS.
      certPath: # [string] Certificate path - Path on server containing certificates
to use. PEM format. Can reference $ENV_VARS.
      caPath: # [string] CA certificate path - Path on server containing CA
certificates to use. PEM format. Can reference $ENV_VARS.
      requestCert: # [boolean] Authenticate client (mutual auth) - Whether to
require clients to present their certificates. Used to perform client authentication
using SSL certs.

      # ------------------------------------------------------

    maxActiveReq: # [number] Max active requests - Maximum number of active requests
per Worker Process. Use 0 for unlimited.
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
is proxied by a device that supports Proxy Protocol V1 or V2.
    captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add
request headers to events, in the __headers field.
    activityLogSampleRate: # [number] Activity log sample rate - How often request
activity is logged at the `info` level. A value of 1 would log every request, 10
every 10th request, etc.
    requestTimeout: # [number] Request timeout (seconds) - How long to wait for an
incoming request to complete before aborting it. Use 0 to disable.
    extractSpans: # [boolean] Extract spans - Toggle to Yes to extract each incoming
span to a separate event.
    extractMetrics: # [boolean] Extract metrics - Toggle to Yes to extract each
incoming Gauge or IntGauge metric to multiple events, one per data point.
    maxActiveCxn: # [number] Max active connections - Maximum number of active
connections allowed per Worker Process, use 0 for unlimited
    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
that references your credentials

    # -------------------------------------------------------


    # -------------- if authType is textSecret ---------------

    textSecret: # [string] Token (text secret) - Select (or create) a stored text
secret

    # -------------------------------------------------------
```

```
    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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
id>.
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # --------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  sqs_input: # [object]
    type: # [string] Input Type
    queueName: # [string] Queue - The name, URL, or ARN of the SQS queue to read
events from. When a non-AWS URL is specified, format must be: '{url}/myQueueName'.
E.g., 'https://host:port/myQueueName'. Value must be a JavaScript expression (which
can evaluate to a constant value), enclosed in quotes or backticks. Can only be
evaluated at init time. E.g. referencing a Global Variable:
`https://host:port/myQueue-${C.vars.myVar}`.
    queueType: # [string] [required] Queue Type - The queue type used (or created).
Defaults to Standard

    # -------------- if queueType is standard ---------------
```

```
    numReceivers: # [number] Num receivers - The Number of receiver processes to
run, the higher the number the better throughput at the expense of CPU overhead

    # ------------------------------------------------------

    awsAccountId: # [string] AWS Account ID - SQS queue owner's AWS account ID.
Leave empty if SQS queue is in same AWS account.
    createQueue: # [boolean] Create Queue - Create queue if it does not exist.
    awsAuthenticationMethod: # [string] Authentication method - AWS authentication
method. Choose Auto to use IAM roles.

    # -------------- 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
references your access key and secret key.

    # ------------------------------------------------------

    awsSecretKey: # [string] Secret key - Secret key
    region: # [string] Region - AWS Region where the SQS queue is located. Required,
unless the Queue entry is a URL or ARN that includes a Region.
    endpoint: # [string] Endpoint - SQS service endpoint. If empty, defaults to AWS'
Region-specific endpoint. Otherwise, it must point to SQS-compatible endpoint.
    signatureVersion: # [string] Signature version - Signature version to use for
signing SQS requests.
    reuseConnections: # [boolean] Reuse connections - Whether to reuse connections
between requests, which can improve performance.
    rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to
reject certificates that cannot be verified against a valid CA (e.g., self-signed
certificates).
    enableAssumeRole: # [boolean] Enable for SQS - Use Assume Role credentials to
access SQS
    assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the
role to assume
    assumeRoleExternalId: # [string] External ID - External ID to use when assuming
role
    maxMessages: # [number] Max Messages - The maximum number of messages SQS should
return in a poll request. Amazon SQS never returns more messages than this value
(however, fewer messages might be returned). Valid values: 1 to 10.
    visibilityTimeout: # [number] Visibility Timeout Seconds - The duration (in
seconds) that the received messages are hidden from subsequent retrieve requests
after being retrieved by a ReceiveMessage request.
    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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    pollTimeout: # [number] Poll timeout (secs) - The amount of time to wait for
events before trying polling again. The lower the number the higher the AWS bill.
The higher the number the longer it will take for the source to react to
configuration changes and system restarts.
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
```

```
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
id>.
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # --------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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
default '0.0.0.0'. For IPv6, enter '::' (all addresses) or specify an IP address.
    udpPort: # [number] UDP port - Enter UDP port number to listen on. Not required
if listening on TCP.
    tcpPort: # [number] TCP port - Enter TCP port number to listen on. Not required
if listening on UDP.
    maxBufferSize: # [number] Max buffer size (events) - Maximum number of events to
buffer when downstream is blocking. Only applies to UDP.
    ipWhitelistRegex: # [string] IP allowlist regex - Regex matching IP addresses
that are allowed to send data
    timestampTimezone: # [string] Default timezone - Timezone to assign to
timestamps without timezone info.
    singleMsgUdpPackets: # [boolean] Single msg per UDP - Whether to treat UDP
packet data received as full syslog message
```

```
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
is proxied by a device that supports Proxy Protocol V1 or V2
    keepFieldsList: # [array of strings] Fields to keep - Wildcard list of fields to
keep from source data, * = ALL (default)
    octetCounting: # [boolean] Octet count framing - Enable if incoming messages use
octet counting per RFC 6587.
    tls: # [object] TLS settings (server side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      privKeyPath: # [string] Private key path - Path on server containing the
private key to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      certPath: # [string] Certificate path - Path on server containing certificates
to use. PEM format. Can reference $ENV_VARS.
      caPath: # [string] CA certificate path - Path on server containing CA
certificates to use. PEM format. Can reference $ENV_VARS.
      requestCert: # [boolean] Authenticate client (mutual auth) - Whether to
require clients to present their certificates. Used to perform client authentication
using SSL certs.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to accept
from connections.
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept
from connections.

      # ----------------------------------------------------------

    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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
```

```
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
id>.
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # ------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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
variables will be resolved, e.g. $CRIBL_HOME/log/.
    depth: # [number] Max depth - Set how many subdirectories deep to search. Use 0
to search only files in the given path, 1 to also look in its immediate
subdirectories, etc. Leave it empty for unlimited depth.

    # ------------------------------------------------------

    interval: # [number] Polling interval - Time, in seconds, between scanning for
files.
    filenames: # [array of strings] Filename allowlist - The full path of discovered
files are matched against this wildcard list.
    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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event
breaking rulesets that will be applied, in order, to the input data stream.
    staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of
time (in milliseconds) the Event Breaker will wait for new data to be sent to a
specific channel, before flushing the data stream out, as-is, to the Pipelines.
    type: # [string] Input Type
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
id>.
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # ------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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
certificate.
      privKeyPath: # [string] Private key path - Path on server containing the
private key to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      certPath: # [string] Certificate path - Path on server containing certificates
to use. PEM format. Can reference $ENV_VARS.
      caPath: # [string] CA certificate path - Path on server containing CA
certificates to use. PEM format. Can reference $ENV_VARS.
      requestCert: # [boolean] Authenticate client (mutual auth) - Whether to
require clients to present their certificates. Used to perform client authentication
using SSL certs.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to accept
from connections.
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept
from connections.

      # ------------------------------------------------------
```

```
    ipWhitelistRegex: # [string] IP Allowlist Regex - Regex matching IP addresses
that are allowed to establish a connection.
    maxActiveCxn: # [number] Max Active Connections - Maximum number of active
connections allowed per Worker Process, use 0 for unlimited
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
is proxied by a device that supports Proxy Protocol V1 or V2.
    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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event
breaking rulesets that will be applied, in order, to the input data stream.
    staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of
time (in milliseconds) the Event Breaker will wait for new data to be sent to a
specific channel, before flushing the data stream out, as-is, to the Pipelines.
    enableHeader: # [boolean] Enable Header - If enabled, client will pass the
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, etc. Header has this format, and
must be followed by a newline: { "authToken" : "myToken", "fields": { "field1":
"value1", "field2": "value2" } }

    # -------------- if enableHeader is true ---------------

    authType: # [string] Authentication method - Enter a token directly, or provide
a secret referencing a token

    # ------------------------------------------------------

    preprocess: # [object]
      disabled: # [boolean] Disabled - Enable Custom Command
      command: # [string] Command - Command to feed the data through (via stdin) and
process its output (stdout)
      args: # [array of strings] Arguments - Arguments
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
```

```
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
id>.
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # ------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  appscope_input: # [object]
    type: # [string] Input Type
    disabled: # [boolean] Disabled - Enable/disable this input
    ipWhitelistRegex: # [string] IP Allowlist Regex - Regex matching IP addresses
that are allowed to establish a connection.
    maxActiveCxn: # [number] Max Active Connections - Maximum number of active
connections allowed per Worker Process, use 0 for unlimited
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
is proxied by a device that supports Proxy Protocol V1 or V2.
    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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event
breaking rulesets that will be applied, in order, to the input data stream.
    staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of
time (in milliseconds) the Event Breaker will wait for new data to be sent to a
specific channel, before flushing the data stream out, as-is, to the Pipelines.
    enableUnixPath: # [boolean] UNIX domain socket - Toggle to Yes to specify a
file-backed UNIX domain socket connection, instead of a network host and port.

    # -------------- 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
certificate.
      privKeyPath: # [string] Private key path - Path on server containing the
private key to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      certPath: # [string] Certificate path - Path on server containing certificates
to use. PEM format. Can reference $ENV_VARS.
      caPath: # [string] CA certificate path - Path on server containing CA
certificates to use. PEM format. Can reference $ENV_VARS.
      requestCert: # [boolean] Authenticate client (mutual auth) - Whether to
require clients to present their certificates. Used to perform client authentication
```

```
using SSL certs.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to accept
from connections.
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept
from connections.

      # ------------------------------------------------------


    # ------------------------------------------------------


    # -------------- if enableUnixPath is true ---------------

    unixSocketPath: # [string] UNIX socket path - Path to the UNIX domain socket to
listen on.
    unixSocketPerms: # [string,number] UNIX socket permissions - Permissions to set
for socket e.g., 777. If empty, falls back to the runtime user's default
permissions.

    # ------------------------------------------------------

    authType: # [string] Authentication method - Enter a token directly, or provide
a secret referencing a token
    authToken: # [string] Auth token - Shared secret to be provided by any client
(in authToken header field). If empty, unauthed access is permitted.

    # -------------- if authType is manual ---------------


    # ------------------------------------------------------

    textSecret: # [string] Auth token (text secret) - Select (or create) a stored
text secret

    # -------------- if authType is secret ---------------


    # ------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------
```

```
    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
id>.
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # ------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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
certificate authentication.
      requestCert: # [boolean] Authenticate client - Required for WEF certificate
authentication.
      certificateName: # [string] Certificate name - Name of the predefined
certificate.
      privKeyPath: # [string] Private key path - Path on server containing the
private key to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      certPath: # [string] [required] Certificate path - Path on server containing
certificates to use. PEM format. Can reference $ENV_VARS.
      caPath: # [string] [required] CA certificate path - Path on server containing
CA certificates to use. PEM format. Can reference $ENV_VARS.
      commonNameRegex: # [string] Common name - Regex matching allowable common
names in peer certificates' subject attribute.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to accept
from connections.
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept
from connections
    maxActiveReq: # [number] Max active requests - Maximum number of active requests
per Worker Process. Use 0 for unlimited.
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
is proxied by a device that supports Proxy Protocol V1 or V2.
    captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add
request headers to events, in the __headers field.
    caFingerprint: # [string] CA fingerprint override - SHA1 fingerprint expected by
the client, if it does not match the first certificate in the configured CA chain
    allowMachineIdMismatch: # [boolean] Allow MachineID mismatch - Allow events to
```

```
be ingested even if their MachineID does not match the client certificate CN.
    subscriptions: # [array] [required] Subscriptions - Subscriptions to events on
forwarding endpoints.
      - subscriptionName: # [string] Name - Friendly name for this subscription.
        version: # [string] Version - Version UUID for this subscription. If any
subscription parameters are modified, this value will change.
        contentFormat: # [string] Format - Content format in which the endpoint
should deliver events.
        heartbeatInterval: # [number] Heartbeat - Max time (in seconds) between
endpoint checkins before considering it unavailable.
        batchTimeout: # [number] Batch timeout - Interval (in seconds) over which
the endpoint should collect events before sending them to Stream.
        readExistingEvents: # [boolean] Read existing events - Set to Yes if a
newly-subscribed endpoint should send previously existing events. Set to No to only
receive new events
        sendBookmarks: # [boolean] Use bookmarks - If toggled to Yes, @{product}
will keep track of which events have been received, resuming from that point after a
re-subscription. This setting takes precedence over 'Read existing events' -- see
the documentation for details.
        compress: # [boolean] Compression - If toggled to Yes, Stream will receive
compressed events from the source.
        targets: # [array of strings] Targets - Enter the DNS names of the endpoints
that should forward these events. You may use wildcards, for example: *.mydomain.com
        querySelector: # [string] Query builder mode - Select the query builder
mode.
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
```

```
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
id>.
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # ------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  win_event_logs_input: # [object]
    type: # [string] Input Type
    logNames: # [array of strings] Event Logs - Enter the event logs to collect. Run
"Get-WinEvent -ListLog *" in PowerShell to see the available logs.
    readMode: # [string] Read Mode - Read all stored and future event logs, or only
future events.
    interval: # [number] Polling Interval - Time, in seconds, between checking for
new entries.
    batchSize: # [number] Batch Size - Maximum number of event records to read in
one interval.
    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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly
to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - 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
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the
filesystem only when it detects backpressure from the processing engine.
WithÂ AlwaysÂ On mode, PQ will always write events directly to the queue before
forwarding them to the processing engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to
hold in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/inputs/<input-
```

```
id>.
      compress: # [string] Compression - Codec to use to compress the persisted
data.

    # ----------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
```

14.3.6. instance.yml

Instance configuration is located under `$CRIBL_HOME/local/_system/instance.yml`.

$CRIBL_HOME/local/_system/instance.yml

```
distributed:
    # mode master | worker | single | edge | managed-edge
  mode: master
  master:
    host: 0.0.0.0
    port: 4203
    tls:
      disabled: true
    ipWhitelistRegex: /.&ast;/
    maxActiveCxn: 0
    authToken: criblmaster
    enabledWorkerRemoteAccess: true
    compression: none
    connectionTimeout: 5000
    writeTimeout: 10000
  group: default
  envRegex: /^CRIBL_/
  tags:
       - tag1
       - tag2
       - tag42
```

14.3.7. jobs.yml

`jobs.yml` maintains parameters for configured Collectors, corresponding to those listed on the UI's **Manage 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
by the same worker node.
  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
Destination that will be used to auto-populate collector settings.
      authType: # [string] Authentication method - Enter authentication data
directly, or select a secret referencing your auth data

      # -------------- if authType is manual ---------------

      connectionString: # [string] Connection string - Enter your Azure storage
account Connection String. If left blank, Cribl Stream will fall back to
env.AZURE_STORAGE_CONNECTION_STRING.

      # -----------------------------------------------------


      # -------------- if authType is secret ---------------

      textSecret: # [string] Connection string (text secret) - Text secret

      # -----------------------------------------------------

      containerName: # [string] Container name - Name of the container to collect
from. This value can be a constant or a JavaScript expression that can only be
evaluated at init time. E.g. referencing a Global Variable:
`myBucket-${C.vars.myVar}`.
      path: # [string] Path - The directory from which to collect data. Templating
is supported, e.g.: myDir/${datacenter}/${host}/${app}/. Time-based tokens are also
supported, e.g.: myOtherDir/${_time:%Y}/${_time:%m}/${_time:%d}/
      extractors: # [array] Path extractors - Allows using template tokens as
context for expressions that enrich discovery results. E.g.: given a template
/path/${epoch}, an extractor under key "epoch" with an expression {date: new
Date(+value*1000)}, will enrich discovery results with a human readable "date" field
        - key: # [string] Token - A token from the template path, e.g.: epoch
          expression: # [string] Extractor expression - JS expression that receives
token under "value" variable, and evaluates to populate event fields, e.g.: {date:
new Date(+value*1000)}
      recurse: # [boolean] Recursive - Whether to recurse through subdirectories.
      maxBatchSize: # [number] Max Batch Size (objects) - Maximum number of metadata
objects to batch before recording as results.

    # -----------------------------------------------------


    # -------------- if type is filesystem ---------------

    conf: # [object] [required]
      outputName: # [string] Auto-populate from - Select a predefined configuration
(e.g., a Destination) to auto-populate collector settings.
      path: # [string] Directory - The directory from which to collect data.
Templating is supported, e.g.: /myDir/${datacenter}/${host}/${app}/. Time-based
tokens are also supported, e.g.: /myOtherDir/${_time:%Y}/${_time:%m}/${_time:%d}/
```

```
      extractors: # [array] Path extractors - Allows using template tokens as
context for expressions that enrich discovery results. E.g.: given a template
/path/${epoch}, an extractor under key "epoch" with an expression {date: new
Date(+value*1000)}, will enrich discovery results with a human readable "date" field
        - key: # [string] Token - A token from the template directory, e.g.: epoch
          expression: # [string] Extractor expression - JS expression that receives
token under "value" variable, and evaluates to populate event fields, e.g.: {date:
new Date(+value*1000)}
      recurse: # [boolean] Recursive - Whether to recurse through subdirectories.
      maxBatchSize: # [number] Max Batch Size (files) - Maximum number of metadata
files to batch before recording as results.

    # -------------------------------------------------------


    # -------------- if type is google_cloud_storage ---------------

    conf: # [object] [required]
      outputName: # [string] Auto-populate from - The name of the predefined
Destination that will be used to auto-populate collector settings.
      bucket: # [string] Bucket name - Name of the bucket to collect from. This
value can be a constant or a JavaScript expression that can only be evaluated at
init time. E.g. referencing a Global Variable: `myBucket-${C.vars.myVar}`.
      path: # [string] Path - The directory from which to collect data. Templating
is supported, e.g.: myDir/${datacenter}/${host}/${app}/. Time-based tokens are also
supported, e.g.: myOtherDir/${_time:%Y}/${_time:%m}/${_time:%d}/
      extractors: # [array] Path extractors - Allows using template tokens as
context for expressions that enrich discovery results. E.g.: given a template
/path/${epoch}, an extractor under key "epoch" with an expression {date: new
Date(+value*1000)}, will enrich discovery results with a human readable "date" field
        - key: # [string] Token - A token from the template path, e.g.: epoch
          expression: # [string] Extractor expression - JS expression that receives
token under "value" variable, and evaluates to populate event fields, e.g.: {date:
new Date(+value*1000)}
      endpoint: # [string] Endpoint - Google Cloud Storage service endpoint. If
empty, the endpoint will be automatically constructed using service credentials.
      disableTimeFilter: # [boolean] Disable time filter - Used to disable collector
event time filtering when a date range is specified.
      recurse: # [boolean] Recursive - Whether to recurse through subdirectories.
      maxBatchSize: # [number] Max Batch Size (objects) - Maximum number of metadata
objects to batch before recording as results.
      authType: # [string] Authentication method - Enter account credentials
manually, or select a secret that references your credentials.

      # -------------- if authType is manual ---------------

      serviceAccountCredentials: # [string] Service account credentials - Contents
of Google Cloud service account credentials (JSON keys) file. To upload a file,
click the upload button at this field's upper right.

      # -------------------------------------------------------


      # -------------- if authType is secret ---------------

      textSecret: # [string] Service account credentials (text secret) - Select (or
create) a stored text secret that references your credentials.

      # -------------------------------------------------------
```

```
    # ------------------------------------------------------


    # -------------- if type is health_check ---------------

    conf: # [object] [required]
      discovery: # [object]
        discoverType: # [string] Discover Type - Defines how task discovery will be
performed. Use None to skip the discovery. Use HTTP Request to make a REST call to
discover tasks. Use Item List to enumerate items for collect to retrieve. Use JSON
Response to manually define discover tasks as a JSON array of objects. Each entry
returned by the discover operation will result in a collect task.

        # -------------- if discoverType is http ---------------

        discoverUrl: # [string] Discover URL - Expression to derive URL to use for
the Discover operation (can be a constant).
        discoverMethod: # [string] Discover method - Discover HTTP method.
        discoverRequestHeaders: # [array] Discover Headers - Optional discover
request headers.
          - name: # [string] Name - Header name.
            value: # [string] Value - JavaScript expression to compute the header
value (can be a constant).
        discoverDataField: # [string] Discover Data Field - Path to field in the
response object which contains discover results (e.g.: level1.name), leave blank if
the result is an array.

        # ------------------------------------------------------


        # -------------- if discoverType is json ---------------

        manualDiscoverResult: # [string] Discover result - Allows hard-coding the
Discover result. Must be a JSON object. Works with the Discover Data field.
        discoverDataField: # [string] Discover data field - Within the response
JSON, name of the field or array element to pull results from. LeaveÂ blank if the
result is an array of values. Sample entry: items, json: { items: [{id: 'first'},
{id: 'second'}] }

        # ------------------------------------------------------


        # -------------- if discoverType is list ---------------

        itemList: # [array of strings] Discover items - Comma-separated list of
items to return from the Discover task. Each item returned will generate a collect
task, and can be referenced using `${id}` in the collect URL, headers, or
parameters.

        # ------------------------------------------------------

      collectUrl: # [string] Health check URL - Expression to derive URL to use for
the health check operation (can be a constant).
      collectMethod: # [string] [required] Health check method - Health check HTTP
method.

      # -------------- if collectMethod is get ---------------

      collectRequestParams: # [array] Health check parameters - Optional health
```

```
check request parameters.
        - name: # [string] Name - Parameter name
          value: # [string] Value - JavaScript expression to compute the parameter
value (can be a constant).

      # ------------------------------------------------------


      # -------------- if collectMethod is post ---------------

      collectRequestParams: # [array] Health check parameters - Optional health
check request parameters.
        - name: # [string] Name - Parameter name.
          value: # [string] Value - JavaScript expression to compute the parameter
value (can be a constant).

      # ------------------------------------------------------


      # -------------- if collectMethod is post_with_body ---------------

      collectBody: # [string] Health check POST Body - Template for POST body to
send with the health check request. You can reference parameters from the Discover
response, using template params of the form: ${variable}.

      # ------------------------------------------------------

      collectRequestHeaders: # [array] Health check headers - Optional health check
request headers.
        - name: # [string] Name - Header Name
          value: # [string] Value - JavaScript expression to compute the header
value (can be a constant).
      authenticateCollect: # [boolean] Authenticate health check - Enable to make
auth health check call.
      authentication: # [string] [required] Authentication - Authentication method
for Discover and Collect REST calls. You can specify API Keyâ□□based authentication
by adding the appropriate CollectÂ headers.

      # -------------- 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
secret that references your credentials

      # ------------------------------------------------------


      # -------------- if authentication is login ---------------

      loginUrl: # [string] Login URL - URL to use for login API call. This call is
expected to be a POST.
      username: # [string] Username - Login username
      password: # [string] Password - Login password
```

```
      loginBody: # [string] POST Body - Template for POST body to send with login
request, ${username} and ${password} are used to specify location of these
attributes in the message
      tokenRespAttribute: # [string] Token Attribute - Path to token attribute in
login response body. Nested attributes are OK.
      authHeaderExpr: # [string] Authorize Expression - JavaScript expression to
compute the Authorization header to pass in discover and collect calls. The value
${token} is used to reference the token obtained from login.
      authRequestHeaders: # [array] Authentication Headers - Optional authentication
request headers.
        - name: # [string] Name - Header name.
          value: # [string] Value - JavaScript expression to compute the header
value (can be a constant).

      # -------------------------------------------------------


      # -------------- if authentication is loginSecret ---------------

      loginUrl: # [string] Login URL - URL to use for login API call, this call is
expected to be a POST.
      credentialsSecret: # [string] Credentials secret - Select (or create) a stored
secret that references your login credentials
      loginBody: # [string] POST Body - Template for POST body to send with login
request, ${username} and ${password} are used to specify location of these
attributes in the message
      tokenRespAttribute: # [string] Token Attribute - Path to token attribute in
login response body. Nested attributes are OK.
      authHeaderExpr: # [string] Authorize Expression - JavaScript expression to
compute the Authorization header to pass in discover and collect calls. The value
${token} is used to reference the token obtained from login.
      authRequestHeaders: # [array] Authentication Headers - Optional authentication
request headers.
        - name: # [string] Name - Header name.
          value: # [string] Value - JavaScript expression to compute the header
value (can be a constant).

      # -------------------------------------------------------


      # -------------- if authentication is oauth ---------------

      loginUrl: # [string] Login URL - URL to use for the OAuth API call. This call
is expected to be a POST.
      tokenRespAttribute: # [string] Token attribute - Path to token attribute in
login response body. Nested attributes are OK.
      authHeaderExpr: # [string] Authorize expression - JavaScript expression to
compute the Authorization header to pass in discover and collect calls. The value
${token} is used to reference the token obtained from login.
      clientSecretParamName: # [string] Client secret parameter - Parameter name
that contains client secret. Defaults to 'client_secret', and is automatically added
to request parameters.
      clientSecretParamValue: # [string] Client secret value - Secret value to add
to HTTP requests as the 'client secret' parameter. Stored on disk encrypted, and is
automatically added to request parameters
      authRequestParams: # [array] Extra authentication parameters - OAuth request
parameters added to the POST body. The Content-Type header will automatically be set
to application/x-www-form-urlencoded.
        - name: # [string] Name - Parameter name.
          value: # [string] Value - JavaScript expression to compute the parameter's
```

```
value, normally enclosed in backticks (e.g.,Â `${earliest}`). IfÂ a constant, use
single quotes (e.g.,Â 'earliest'). ValuesÂ without delimiters (e.g.,Â earliest) are
evaluated as strings.
      authRequestHeaders: # [array] Authentication headers - Optional authentication
request headers.
        - name: # [string] Name - Header name.
          value: # [string] Value - JavaScript expression to compute the header's
value, normally enclosed in backticks (e.g.,Â `${earliest}`). IfÂ a constant, use
single quotes (e.g.,Â 'earliest'). ValuesÂ without delimiters (e.g.,Â earliest) are
evaluated as strings.

      # ------------------------------------------------------


      # -------------- if authentication is oauthSecret ---------------

      loginUrl: # [string] Login URL - URL to use for the OAuth API call. This call
is expected to be a POST.
      tokenRespAttribute: # [string] Token attribute - Path to token attribute in
login response body. Nested attributes are OK.
      authHeaderExpr: # [string] Authorize expression - JavaScript expression to
compute the Authorization header to pass in discover and collect calls. The value
${token} is used to reference the token obtained from login.
      clientSecretParamName: # [string] Client secret parameter - Parameter name
that contains client secret. Defaults to 'client_secret', and is automatically added
to request parameters.
      textSecret: # [string] Client secret value (text secret) - Select (or create)
a text secret that contains the client secret's value.
      authRequestParams: # [array] Extra authentication parameters - OAuth request
parameters added to the POST body. The Content-Type header will automatically be set
to application/x-www-form-urlencoded.
        - name: # [string] Name - Parameter name.
          value: # [string] Value - JavaScript expression to compute the parameter's
value, normally enclosed in backticks (e.g.,Â `${earliest}`). IfÂ a constant, use
single quotes (e.g.,Â 'earliest'). ValuesÂ without delimiters (e.g.,Â earliest) are
evaluated as strings.
      authRequestHeaders: # [array] Authentication headers - Optional authentication
request headers.
        - name: # [string] Name - Header name.
          value: # [string] Value - JavaScript expression to compute the header's
value, normally enclosed in backticks (e.g.,Â `${earliest}`). IfÂ a constant, use
single quotes (e.g.,Â 'earliest'). ValuesÂ without delimiters (e.g.,Â earliest) are
evaluated as strings.

      # ------------------------------------------------------

      timeout: # [number] Request Timeout (secs) - HTTP request inactivity timeout,
use 0 to disable
      defaultBreakers: # [string] Hidden Default Breakers
      safeHeaders: # [array of strings] Safe headers - List of headers that are safe
to log in plain text.

    # ------------------------------------------------------


    # -------------- if type is office365_mgmt ---------------

    conf: # [object] [required]
      plan_type: # [string] Subscription plan - Office 365 subscription plan for
your organization, typically Enterprise
```

```
      tenant_id: # [string] [required] Tenant identifier - Directory ID (tenant
identifier) in Azure Active Directory
      app_id: # [string] [required] Application identifier - Identifier of the
registered application in Azure Active Directory.
      client_secret: # [string] [required] Client secret - Application key of the
registered application.
      publisher_identifier: # [string] Publisher identifier - Optional
PublisherIdentifier to use in API requests, defaults to tenant id if not defined.
      content_type: # [string] [required] Content type - The type of content to
retrieve from the Office 365 management communications API.

    # ------------------------------------------------------


    # -------------- if type is office365_service ---------------

    conf: # [object] [required]
      tenant_id: # [string] Tenant identifier - Directory ID (tenant identifier) in
Azure Active Directory
      app_id: # [string] [required] Application identifier - Identifier of the
registered application in Azure Active Directory.
      client_secret: # [string] [required] Client secret - Application key of the
registered application.
      content_type: # [string] [required] Content type - The type of content to
retrieve from the Office 365 service communications API.

    # ------------------------------------------------------


    # -------------- if type is prometheus ---------------

    conf: # [object] [required]
      dimensionList: # [array] Extra Dimensions - Other dimensions to include in
events
      username: # [string] Username - Optional username for Basic authentication
      password: # [string] Password - Optional password for Basic authentication
      discoveryType: # [string] Discovery Type - Target discovery mechanism, use
static to manually enter a list of targets

      # -------------- if discoveryType is static ---------------

      targetList: # [array] Targets - List of Prometheus targets to pull metrics
from, values can be in URL or host[:port] format. For example:
http://localhost:9090/metrics, localhost:9090, or localhost. In the cases where just
host[:port] are specified, the endpoint will resolve to
'http://host[:port]/metrics'.

      # ------------------------------------------------------


      # -------------- 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
metrics
      scrapePath: # [string] Metrics Path - Path to use when collecting metrics from
discovered targets

      # ------------------------------------------------------
```

```
      # -------------- if discoveryType is ec2 ---------------

      usePublicIp: # [boolean] Use Public IP - Use public IP address for discovered
targets. Set to false if the private IP address should be used.
      scrapeProtocol: # [string] Metrics Protocol - Protocol to use when collecting
metrics
      scrapePort: # [number] Metrics Port - The port number in the metrics URL for
discovered targets.
      scrapePath: # [string] Metrics Path - Path to use when collecting metrics from
discovered targets
      searchFilter: # [array] Search Filter - EC2 Instance Search Filter
        - Name: # [string] Filter Name - Search filter attribute name, see:
https://docs.aws.amazon.com/AWSEC2/latest/APIReference/API_DescribeInstances.html
for more information. Attributes can be manually entered if not present in the drop
down list
          Values: # [array of strings] Filter Values - Search Filter Values
      region: # [string] Region - Region from which to retrieve data.
      awsAuthenticationMethod: # [string] Authentication Method - AWS authentication
method
      enableAssumeRole: # [boolean] Enable Assume Role - Use Assume Role credentials
      endpoint: # [string] Endpoint - EC2 service endpoint. If empty, defaults to
AWS' Region-specific endpoint. Otherwise, used to point to a EC2-compatible
endpoint.
      signatureVersion: # [string] Signature version - Signature version to use for
signing EC2 requests

      # ------------------------------------------------------


    # ------------------------------------------------------


    # -------------- if type is rest ---------------

    conf: # [object] [required]
      discovery: # [object]
        discoverType: # [string] Discover type - Defines how task discovery will be
performed. Use None to skip the discovery. Use HTTP Request to make a REST call to
discover tasks. Use Item List to enumerate items for collect to retrieve. Use JSON
Response to manually define discover tasks as a JSON array of objects. Each entry
returned by the discover operation will result in a collect task.

        # -------------- if discoverType is http ---------------

        discoverUrl: # [string] Discover URL - Expression to derive URL to use for
the Discover operation (can be a constant).
        discoverMethod: # [string] Discover method - Discover HTTP method.
        discoverRequestHeaders: # [array] Discover headers - Optional discover
request headers.
          - name: # [string] Name - Header name.
            value: # [string] Value - JavaScript expression to compute the
parameter's value, normally enclosed in backticks (e.g.,Â `${earliest}`). IfÂ a
constant, use single quotes (e.g.,Â 'earliest'). ValuesÂ without delimiters
(e.g.,Â earliest) are evaluated as strings.
        discoverDataField: # [string] Discover data field - Path to field in the
response object which contains discover results (e.g.: level1.name), leave blank if
the result is an array.
```

```
        # ------------------------------------------------------


        # -------------- if discoverType is json ---------------

        manualDiscoverResult: # [string] Discover result - Allows hard-coding the
Discover result. Must be a JSON object. Works with the Discover Data field.
        discoverDataField: # [string] Discover data field - Within the response
JSON, name of the field or array element to pull results from. LeaveÂ blank if the
result is an array of values. Sample entry: items, json: { items: [{id: 'first'},
{id: 'second'}] }

        # ------------------------------------------------------


        # -------------- if discoverType is list ---------------

        itemList: # [array of strings] Discover items - Comma-separated list of
items to return from the Discover task. Each item returned will generate a collect
task, and can be referenced using `${id}` in the collect URL, headers, or
parameters.

        # ------------------------------------------------------

      collectUrl: # [string] Collect URL - Expression to derive URL to use for the
collect operation (can be a constant).
      collectMethod: # [string] [required] Collect method - Collect HTTP method.

      # -------------- if collectMethod is get ---------------

      collectRequestParams: # [array] Collect parameters - Optional collect request
parameters.
        - name: # [string] Name - Parameter name
          value: # [string] Value - JavaScript expression to compute the parameter's
value, normally enclosed in backticks (e.g.,Â `${earliest}`). IfÂ a constant, use
single quotes (e.g.,Â 'earliest'). ValuesÂ without delimiters (e.g.,Â earliest) are
evaluated as strings.

      # ------------------------------------------------------


      # -------------- if collectMethod is post ---------------

      collectRequestParams: # [array] Collect parameters - Optional collect request
parameters.
        - name: # [string] Name - Parameter name.
          value: # [string] Value - JavaScript expression to compute the parameter's
value, normally enclosed in backticks (e.g.,Â `${earliest}`). IfÂ a constant, use
single quotes (e.g.,Â 'earliest'). ValuesÂ without delimiters (e.g.,Â earliest) are
evaluated as strings.

      # ------------------------------------------------------


      # -------------- if collectMethod is post_with_body ---------------

      collectBody: # [string] Collect POST body - Template for POST body to send
with the Collect request. Reference global variables, functions, or parameters from
the Discover response using template params: `${C.vars.myVar}`, or `${Date.now()}`,
`${param}`.
```

```
      # ------------------------------------------------------


      # -------------- if collectMethod is other ---------------

      collectVerb: # [string] Collect verb - DIY HTTP verb to use for the collect
operation.
      collectBody: # [string] Collect body - Template for body to send with the
Collect request. Reference global variables, functions, or parameters from the
Discover response using template params: `${C.vars.myVar}`, or `${Date.now()}`,
`${param}`
      collectRequestParams: # [array] Collect parameters - Optional collect request
parameters.
        - name: # [string] Name - Parameter name.
          value: # [string] Value - JavaScript expression to compute the parameter's
value, normally enclosed in backticks (e.g.,Â `${earliest}`). IfÂ a constant, use
single quotes (e.g.,Â 'earliest'). ValuesÂ without delimiters (e.g.,Â earliest) are
evaluated as strings.

      # ------------------------------------------------------

      collectRequestHeaders: # [array] Collect headers - Optional collect request
headers.
        - name: # [string] Name - Header Name
          value: # [string] Value - JavaScript expression to compute the header's
value, normally enclosed in backticks (e.g.,Â `${earliest}`). IfÂ a constant, use
single quotes (e.g.,Â 'earliest'). ValuesÂ without delimiters (e.g.,Â earliest) are
evaluated as strings.
      pagination: # [object]
        type: # [string] Pagination - Select collect pagination scheme

        # -------------- if type is response_body ---------------

        attribute: # [array,string] Response attribute - The name of the attribute
in the response that contains next page information
        maxPages: # [number] Max pages - The maximum number of pages to retrieve,
set to 0 to retrieve all pages

        # ------------------------------------------------------


        # -------------- if type is response_header_link ---------------

        nextRelationAttribute: # [string] Next page relation name - Relation name
used in the link header that refers to the next page in the result set. In this
example link header, rel="next" to the next page of results:
<https://myHost/curPage>; rel="self" <https://myHost/nextPage>; rel="next"
        curRelationAttribute: # [string] Current page relation name - Optional
relation name used in the link header that refers to the current result set. In this
example link header, rel="self" refers to the current page of results:
<https://myHost/curPage>; rel="self" <https://myHost/nextPage>; rel="next"
        maxPages: # [number] Max pages - The maximum number of pages to retrieve,
set to 0 to retrieve all pages

        # ------------------------------------------------------


        # -------------- if type is request_offset ---------------
```

```
        offsetField: # [string] Offset field name - Query string parameter that sets
the index from which to begin returning records. E.g.: /api/v1/query?
term=cribl&limit=100&offset=0
        offset: # [number] Starting offset - Offset index from which to start
request. Defaults to undefined, which will start collection from the first record.
        offsetSpacer: # [null]
        limitField: # [string] Limit field name - Query string parameter to set the
number of records retrieved per request. E.g.: /api/v1/query?
term=cribl&limit=100&offset=0
        limit: # [number] Limit - Maximum number of records to collect per request.
        limitSpacer: # [null]
        totalRecordField: # [string] Total record count field name - Identifies the
attribute in the response that contains the total number of records for the query.
        maxPages: # [number] Max pages - The maximum number of pages to retrieve.
Set to 0 to retrieve all pages.
        zeroIndexed: # [boolean] Zero-based index - Toggle to Yes to indicate that
the first record in the requested data is at index 0. The default (No) indicates
index 1.

        # ------------------------------------------------------


        # -------------- if type is request_page ---------------

        pageField: # [string] Page number field name - Query string parameter that
sets the page index to be returned. E.g.: /api/v1/query?
term=cribl&page_size=100&page_number=0
        page: # [number] Starting page number - Page number from which to start
request. Defaults to undefined, which will start collection from the first page.
        offsetSpacer: # [null]
        sizeField: # [string] Page size field name - Query string parameter to set
the number of records retrieved per request. E.g.: /api/v1/query?
term=cribl&page_size=100&page_number=0
        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
attribute in the response that contains the total number of pages for the query.
        totalRecordField: # [string] Total record count field name - Identifies the
attribute in the response that contains the total number of records for the query.
        maxPages: # [number] Max pages - The maximum number of pages to retrieve.
Set to 0 to retrieve all pages.
        zeroIndexed: # [boolean] zero-based index - Toggle to Yes to indicate that
the first page in the requested data is at index 0. The default (No) indicates index
1.

        # ------------------------------------------------------

      authentication: # [string] [required] Authentication - Authentication method
for Discover and Collect REST calls. You can specify API Keyâ□□based authentication
by adding the appropriate CollectÂ headers.

      # -------------- 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
secret that references your credentials

      # ------------------------------------------------------


      # -------------- if authentication is login ---------------

      loginUrl: # [string] Login URL - URL to use for login API call. This call is
expected to be a POST.
      username: # [string] Username - Login username
      password: # [string] Password - Login password
      loginBody: # [string] POST body - Template for POST body to send with login
request, ${username} and ${password} are used to specify location of these
attributes in the message
      tokenRespAttribute: # [string] Token attribute - Path to token attribute in
login response body. Nested attributes are OK.
      authHeaderExpr: # [string] Authorize expression - JavaScript expression to
compute the Authorization header to pass in discover and collect calls. The value
${token} is used to reference the token obtained from login.
      authRequestHeaders: # [array] Authentication headers - Optional authentication
request headers.
        - name: # [string] Name - Header name.
          value: # [string] Value - JavaScript expression to compute the header's
value, normally enclosed in backticks (e.g.,Â `${earliest}`). IfÂ a constant, use
single quotes (e.g.,Â 'earliest'). ValuesÂ without delimiters (e.g.,Â earliest) are
evaluated as strings.

      # ------------------------------------------------------


      # -------------- if authentication is loginSecret ---------------

      loginUrl: # [string] Login URL - URL to use for login API call, this call is
expected to be a POST.
      credentialsSecret: # [string] Credentials secret - Select (or create) a stored
secret that references your login credentials
      loginBody: # [string] POST body - Template for POST body to send with login
request, ${username} and ${password} are used to specify location of these
attributes in the message
      tokenRespAttribute: # [string] Token attribute - Path to token attribute in
login response body. Nested attributes are OK.
      authHeaderExpr: # [string] Authorize expression - JavaScript expression to
compute the Authorization header to pass in discover and collect calls. The value
${token} is used to reference the token obtained from login.
      authRequestHeaders: # [array] Authentication headers - Optional authentication
request headers.
        - name: # [string] Name - Header name.
          value: # [string] Value - JavaScript expression to compute the parameter's
value, normally enclosed in backticks (e.g.,Â `${earliest}`). IfÂ a constant, use
single quotes (e.g.,Â 'earliest'). ValuesÂ without delimiters (e.g.,Â earliest) are
evaluated as strings.

      # ------------------------------------------------------


      # -------------- if authentication is oauth ---------------

      loginUrl: # [string] Login URL - URL to use for the OAuth API call. This call
```

```
is expected to be a POST.
      tokenRespAttribute: # [string] Token attribute - Path to token attribute in
login response body. Nested attributes are OK.
      authHeaderExpr: # [string] Authorize expression - JavaScript expression to
compute the Authorization header to pass in discover and collect calls. The value
${token} is used to reference the token obtained from login.
      clientSecretParamName: # [string] Client secret parameter - Parameter name
that contains client secret. Defaults to 'client_secret', and is automatically added
to request parameters.
      clientSecretParamValue: # [string] Client secret value - Secret value to add
to HTTP requests as the 'client secret' parameter. Stored on disk encrypted, and is
automatically added to request parameters
      authRequestParams: # [array] Extra authentication parameters - OAuth request
parameters added to the POST body. The Content-Type header will automatically be set
to application/x-www-form-urlencoded.
        - name: # [string] Name - Parameter name.
          value: # [string] Value - JavaScript expression to compute the parameter's
value, normally enclosed in backticks (e.g.,Â `${earliest}`). IfÂ a constant, use
single quotes (e.g.,Â 'earliest'). ValuesÂ without delimiters (e.g.,Â earliest) are
evaluated as strings.
      authRequestHeaders: # [array] Authentication headers - Optional authentication
request headers.
        - name: # [string] Name - Header name.
          value: # [string] Value - JavaScript expression to compute the header's
value, normally enclosed in backticks (e.g.,Â `${earliest}`). IfÂ a constant, use
single quotes (e.g.,Â 'earliest'). ValuesÂ without delimiters (e.g.,Â earliest) are
evaluated as strings.

      # ------------------------------------------------------


      # -------------- if authentication is oauthSecret ---------------

      loginUrl: # [string] Login URL - URL to use for the OAuth API call. This call
is expected to be a POST.
      tokenRespAttribute: # [string] Token attribute - Path to token attribute in
login response body. Nested attributes are OK.
      authHeaderExpr: # [string] Authorize expression - JavaScript expression to
compute the Authorization header to pass in discover and collect calls. The value
${token} is used to reference the token obtained from login.
      clientSecretParamName: # [string] Client secret parameter - Parameter name
that contains client secret. Defaults to 'client_secret', and is automatically added
to request parameters.
      textSecret: # [string] Client secret value (text secret) - Select (or create)
a text secret that contains the client secret's value.
      authRequestParams: # [array] Extra authentication parameters - OAuth request
parameters added to the POST body. The Content-Type header will automatically be set
to application/x-www-form-urlencoded.
        - name: # [string] Name - Parameter name.
          value: # [string] Value - JavaScript expression to compute the parameter's
value, normally enclosed in backticks (e.g.,Â `${earliest}`). IfÂ a constant, use
single quotes (e.g.,Â 'earliest'). ValuesÂ without delimiters (e.g.,Â earliest) are
evaluated as strings.
      authRequestHeaders: # [array] Authentication headers - Optional authentication
request headers.
        - name: # [string] Name - Header name.
          value: # [string] Value - JavaScript expression to compute the header's
value, normally enclosed in backticks (e.g.,Â `${earliest}`). IfÂ a constant, use
single quotes (e.g.,Â 'earliest'). ValuesÂ without delimiters (e.g.,Â earliest) are
evaluated as strings.
```

```
      # ------------------------------------------------------

      timeout: # [number] Request timeout (secs) - HTTP request inactivity timeout,
use 0 to disable
      useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS
lookup. Suitable when DNS server returns multiple addresses in sort order.
      disableTimeFilter: # [boolean] Disable time filter - Used to disable collector
event time filtering when a date range is specified.
      safeHeaders: # [array of strings] Safe headers - List of headers that are safe
to log in plain text.

    # ------------------------------------------------------


    # -------------- if type is s3 ---------------

    conf: # [object] [required]
      outputName: # [string] Auto-populate from - The name of the predefined
Destination that will be used to auto-populate collector settings.
      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 supported, e.g.: myDir/${datacenter}/${host}/${app}/. Time-based tokens are also
supported, e.g.: myOtherDir/${_time:%Y}/${_time:%m}/${_time:%d}/
      extractors: # [array] Path extractors - Allows using template tokens as
context for expressions that enrich discovery results. E.g.: given a template
/path/${epoch}, an extractor under key "epoch" with an expression {date: new
Date(+value*1000)}, will enrich discovery results with a human readable "date" field
        - key: # [string] Token - A token from the template path, e.g.: epoch
          expression: # [string] Extractor expression - JS expression that receives
token under "value" variable, and evaluates to populate event fields, e.g.: {date:
new Date(+value*1000)}
      awsAuthenticationMethod: # [string] Authentication Method - AWS authentication
method. Choose Auto to use IAM roles.

      # -------------- if awsAuthenticationMethod is manual ---------------

      awsApiKey: # [string] Access key - Access key. If not present, will fall back
to env.AWS_ACCESS_KEY_ID, or to the metadata endpoint for IAM creds. Optional when
running on AWS.
      awsSecretKey: # [string] Secret key - Secret key. If not present, will fall
back to env.AWS_SECRET_ACCESS_KEY, or to the metadata endpoint for IAM creds.
Optional when running on AWS.

      # ------------------------------------------------------


      # -------------- if awsAuthenticationMethod is secret ---------------

      awsSecret: # [string] Secret key pair - Select (or create) a stored secret
that references AWS access key and secret key.

      # ------------------------------------------------------

      endpoint: # [string] Endpoint - S3 service endpoint. If empty, the endpoint
will be automatically constructed from the region.
      signatureVersion: # [string] Signature version - Signature version to use for
signing S3 requests.
      enableAssumeRole: # [boolean] Enable Assume Role - Use Assume Role
```

```
credentials.
      assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the
role to assume.
      assumeRoleExternalId: # [string] External ID - External ID to use when
assuming role.
      recurse: # [boolean] Recursive - Whether to recurse through subdirectories.
      maxBatchSize: # [number] Max Batch Size (objects) - Maximum number of metadata
objects to batch before recording as results.
      reuseConnections: # [boolean] Reuse Connections - Whether to reuse connections
between requests, which can improve performance.
      rejectUnauthorized: # [boolean] Reject Unauthorized Certificates - Whether to
reject certificates that cannot be verified against a valid CA (e.g., self-signed
certificates).
      verifyPermissions: # [boolean] Verify bucket permissions - Disable if you can
access files within the bucket but not the bucket itself. Resolves errors of the
form "discover task initialization failed...error: Forbidden".

    # ------------------------------------------------------


    # -------------- if type is script ---------------

    conf: # [object] [required]
      discoverScript: # [string] Discover Script - Script to discover what to
collect. Should output one task per line in stdout.
      collectScript: # [string] [required] Collect Script - Script to run to perform
data collections. Task passed in as $CRIBL_COLLECT_ARG. Should output results to
stdout.
      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
expression, default is https://localhost:8089.
      search: # [string] Search - Enter Splunk search here. For example:
'index=myAppLogs level=error channel=myApp' OR '| mstats avg(myStat) as myStat WHERE
index=myStatsIndex.'
      earliest: # [string] Earliest - The earliest time boundary for the search. Can
be an exact or relative time. For example: '2022-01-14T12:00:00Z' or '-16m@m'
      latest: # [string] Latest - The latest time boundary for the search. Can be an
exact or relative time. For example: '2022-01-14T12:00:00Z' or '-1m@m'
      endpoint: # [string] [required] Search endpoint - REST API used to create a
search.
      outputMode: # [string] [required] Output mode - Format of the returned output
      collectRequestParams: # [array] Extra parameters - Optional collect request
parameters.
        - name: # [string] Name - Parameter name
          value: # [string] Value - JavaScript expression to compute the parameter's
value, normally enclosed in backticks (e.g.,Â `${earliest}`). IfÂ a constant, use
single quotes (e.g.,Â 'earliest'). ValuesÂ without delimiters (e.g.,Â earliest) are
evaluated as strings.
      collectRequestHeaders: # [array] Extra headers - Optional collect request
headers.
        - name: # [string] Name - Header Name
          value: # [string] Value - JavaScript expression to compute the header's
value, normally enclosed in backticks (e.g.,Â `${earliest}`). IfÂ a constant, use
```

```
single quotes (e.g.,Â 'earliest'). ValuesÂ without delimiters (e.g.,Â earliest) are
evaluated as strings.
      authentication: # [string] [required] Authentication - Authentication method
for Discover and Collect REST calls.

      # -------------- 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
secret that references your credentials

      # ----------------------------------------------------------


      # -------------- if authentication is login ---------------

      loginUrl: # [string] Login URL - URL to use for login API call. This call is
expected to be a POST.
      username: # [string] Username - Login username
      password: # [string] Password - Login password
      loginBody: # [string] POST Body - Template for POST body to send with login
request, ${username} and ${password} are used to specify location of these
attributes in the message
      tokenRespAttribute: # [string] Token Attribute - Path to token attribute in
login response body. Nested attributes are OK.
      authHeaderExpr: # [string] Authorize Expression - JavaScript expression to
compute the Authorization header to pass in discover and collect calls. The value
${token} is used to reference the token obtained from login.

      # ----------------------------------------------------------


      # -------------- if authentication is loginSecret ---------------

      loginUrl: # [string] Login URL - URL to use for login API call, this call is
expected to be a POST.
      credentialsSecret: # [string] Credentials secret - Select (or create) a stored
secret that references your login credentials
      loginBody: # [string] POST Body - Template for POST body to send with login
request, ${username} and ${password} are used to specify location of these
attributes in the message
      tokenRespAttribute: # [string] Token Attribute - Path to token attribute in
login response body. Nested attributes are OK.
      authHeaderExpr: # [string] Authorize Expression - JavaScript expression to
compute the Authorization header to pass in discover and collect calls. The value
${token} is used to reference the token obtained from login.

      # ----------------------------------------------------------

      timeout: # [number] Request Timeout (secs) - HTTP request inactivity timeout,
use 0 to disable
      useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS
lookup. Suitable when DNS server returns multiple addresses in sort order.
```

```
      disableTimeFilter: # [boolean] Disable time filter - Used to disable collector
event time filtering when a date range is specified.

    # -------------------------------------------------------

    destructive: # [boolean] Destructive - If set to Yes, the collector will delete
any files that it collects (where applicable).
  input: # [object]
    type: # [string]
    breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event
breaking rulesets that will be applied, in order, to the input data stream.
    staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of
time (in milliseconds) the Event Breaker will wait for new data to be sent to a
specific channel, before flushing the data stream out, as-is, to the Pipelines.
    sendToRoutes: # [boolean] Send to Routes - If set to Yes, events will be sent to
normal routing and event processing. Set to No if you want to select a specific
Pipeline/Destination combination.

    # -------------- if sendToRoutes is true ---------------

    pipeline: # [string] Pre-Processing Pipeline - Pipeline to process results
before sending to routes. Optional.

    # -------------------------------------------------------


    # -------------- 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
process its output (stdout)
      args: # [array of strings] Arguments - Arguments
    throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to
throttle while writing to an output. Also takes values with multiple-byte units,
such as KB, MB, GB, etc. (E.g., 42 MB.) Default value of 0 specifies no throttling.
    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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
  type: # [string] Job type - Job type.
  ttl: # [string] Time to live - Time to keep the job's artifacts on disk after job
completion. This also affects how long a job is listed in the Job Inspector.
  removeFields: # [array of strings] Remove Discover fields - List of fields to
remove from Discover results. Wildcards (e.g.: aws*) are allowed. This is useful
when discovery returns sensitive fields that should not be exposed in the Jobs user
interface.
  resumeOnBoot: # [boolean] Resume job on boot - Resumes the ad hoc job if a failure
condition causes Stream to restart during job execution.
  environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
  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
job.
    maxConcurrentRuns: # [number] Max concurrent runs - The maximum number of
instances that may be running of this scheduled job at any given time.
    skippable: # [boolean] Skippable - Skippable jobs can be delayed, up to their
next run time, if the system is hitting concurrency limits.
    run: # [object] Run settings

      # -------------- if type is collection ---------------

      rescheduleDroppedTasks: # [boolean] Reschedule tasks - Reschedule tasks that
failed with non-fatal errors.
      maxTaskReschedule: # [number] Max task reschedule - Max number of times a task
can be rescheduled.
      logLevel: # [string] Log Level - Level at which to set task logging.
      jobTimeout: # [string] Job timeout - Maximum time the job is allowed to run
(e.g., 30, 45s or 15m). Units are seconds, if not specified. Enter 0 for unlimited
time.
      mode: # [string] Mode - Job run mode. Preview will either return up to N
matching results, or will run until capture time T is reached. Discovery will gather
the list of files to turn into streaming tasks, without running the data collection
job. Full Run will run the collection job.

      # -------------- if mode is list ---------------

      discoverToRoutes: # [boolean] Send to Routes - If true, send discover results
to routes

      # -----------------------------------------------------


      # -------------- if mode is preview ---------------

      capture: # [object] Capture Settings
        duration: # [number] Capture Time (sec) - Amount of time to keep capture
open, in seconds.
        maxEvents: # [number] Capture Up to N Events - Maximum number of events to
capture.
        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 Latest times (defaults to UTC).
      expression: # [string] Filter - A filter for tokens in the provided collect
path and/or the events being collected
      minTaskSize: # [string] Lower task bundle size - Limits the bundle size for
small tasks. E.g., bundle five 200KB files into one 1M task.
      maxTaskSize: # [string] Upper task bundle size - Limits the bundle size for
files above the Lower task bundle size. E.g., bundle five 2MB files into one 10MB
task bundle. Files greater than this size will be assigned to individual tasks.

    # -----------------------------------------------------

  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
```

```
Stream.
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
write task results to disk.
    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
completion. This also affects how long a job is listed in the Job Inspector.
  removeFields: # [array of strings] Remove Discover fields - List of fields to
remove from Discover results. Wildcards (e.g.: aws*) are allowed. This is useful
when discovery returns sensitive fields that should not be exposed in the Jobs user
interface.
  resumeOnBoot: # [boolean] Resume job on boot - Resumes the ad hoc job if a failure
condition causes Stream to restart during job execution.
  environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
  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
job.
    maxConcurrentRuns: # [number] Max concurrent runs - The maximum number of
instances that may be running of this scheduled job at any given time.
    skippable: # [boolean] Skippable - Skippable jobs can be delayed, up to their
next run time, if the system is hitting concurrency limits.
    run: # [object] Run settings

      # -------------- if type is collection ---------------

      rescheduleDroppedTasks: # [boolean] Reschedule tasks - Reschedule tasks that
failed with non-fatal errors.
      maxTaskReschedule: # [number] Max task reschedule - Max number of times a task
can be rescheduled.
      logLevel: # [string] Log Level - Level at which to set task logging.
      jobTimeout: # [string] Job timeout - Maximum time the job is allowed to run
(e.g., 30, 45s or 15m). Units are seconds, if not specified. Enter 0 for unlimited
time.
      mode: # [string] Mode - Job run mode. Preview will either return up to N
matching results, or will run until capture time T is reached. Discovery will gather
the list of files to turn into streaming tasks, without running the data collection
job. Full Run will run the collection job.

      # -------------- if mode is list ---------------

      discoverToRoutes: # [boolean] Send to Routes - If true, send discover results
to routes

      # ------------------------------------------------------


      # -------------- if mode is preview ---------------

      capture: # [object] Capture Settings
        duration: # [number] Capture Time (sec) - Amount of time to keep capture
open, in seconds.
```

```
        maxEvents: # [number] Capture Up to N Events - Maximum number of events to
capture.
        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 Latest times (defaults to UTC).
      expression: # [string] Filter - A filter for tokens in the provided collect
path and/or the events being collected
      minTaskSize: # [string] Lower task bundle size - Limits the bundle size for
small tasks. E.g., bundle five 200KB files into one 1M task.
      maxTaskSize: # [string] Upper task bundle size - Limits the bundle size for
files above the Lower task bundle size. E.g., bundle five 2MB files into one 10MB
task bundle. Files greater than this size will be assigned to individual tasks.

    # ------------------------------------------------------

  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
```

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.

14.3.8. job-limits.yml

`job-limits.yml` maintains parameters for collection jobs and system tasks. In the UI, you can configure these at **Group Settings** > **General Settings** > **Limits** > **Jobs**.

$CRIBL_HOME/default/cribl/job-limits.yml

```
concurrentJobLimit: # [number] Concurrent job limit - The total number of jobs that may run concurrently
concurrentSystemJobLimit: # [number] Concurrent system job limit - The total number of system jobs that may run concurrently
concurrentScheduledJobLimit: # [number] Concurrent scheduled job limit - The total number of scheduled jobs that may run concurrently. Limit is relative to concurrent job limit.
concurrentTaskLimit: # [number] Concurrent task limit - The total number of tasks that a worker process may run concurrently
concurrentSystemTaskLimit: # [number] Concurrent system task limit - The number of system tasks that a worker process may run concurrently
maxTaskPerc: # [number] Max task usage percentage - Value from 0 to 1 representing the percentage of total tasks within the system a job may consume
taskPollTimeoutMs: # [number] Task poll timeout - The number of milliseconds the worker's task handler will wait to receive a task before retrying the request for a task.
jobArtifactsReaperPeriod: # [string] Artifact reaper period - Time period at which the system attempts to reap stale disk artifacts belonging to the jobs
finishedJobArtifactsLimit: # [number] Finished job artifacts limit - Maximum number of finished job artifacts to keep on disk.
finishedTaskArtifactsLimit: # [number] Finished task artifacts limit - Maximum number of finished task artifacts to keep on disk per job on each worker node.
taskManifestFlushPeriodMs: # [number] Manifest flush period - The rate at which a job's task manifest should be refreshed in milliseconds.
taskManifestMaxBufferSize: # [number] Manifest max buffer size - The maximum number of tasks the task manifest can hold in memory before flushing to disk.
taskManifestReadBufferSize: # [string] Manifest reader buffer size - The number of bytes the task manifest reader should pull from disk.
schedulingPolicy: # [string] Job dispatching - The method by which tasks are assigned to worker processes to complete the work
jobTimeout: # [string] 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.
taskHeartbeatPeriod: # [number] Task heartbeat period - The heartbeat period (in seconds) for tasks to report back to the leader/API.
```

14.3.9. licenses.yml

`licenses.yml` maintains a list of Cribl Edge licenses.

$CRIBL_HOME/default/cribl/licenses.yml

```
licenses: # [array of string] - list of licenses
```

14.3.10. limits.yml

`limits.yml` maintains parameters for the system. In a distributed deployment, these parameters are configured on both the Leader and the Workers.

In the UI, you can configure them:

- On the Leader at **Settings** > **Global Settings** > **General Settings > Limits**.
- On a single instance at **Settings** > **General Settings > Limits**.
- On Worker Groups at `<group-name>` > **Fleet Settings** > **General Settings > Limits**.

$CRIBL_HOME/default/cribl/limits.yml

```
samples: # [object] Samples
  maxSize: # [string] Max sample size - Maximum file size for the sample, in binary
units (KB, MB). (Max. 3MB.)
minFreeSpace: # [string] Min free disk space - The minimum amount of disk space in
the system before various features will take measures to prevent disk usage (KB, MB,
etc.).
metricsGCPeriod: # [string] Metrics GC period - The interval on which the system
attempts to free memory, by pruning stale metrics from the Stream system metrics
store.
metricsMaxCardinality: # [number] Metrics cardinality limit - The system's allowed
number of permutations of a given metric name.
metricsMaxDiskSpace: # [string] Metrics max disk space - Maximum allowed disk space
for persisting metrics to disk.
metricsWorkerIdBlacklist: # [array of strings] Metrics worker tracking - List of
metric names for which to disable tracking of Worker Node ID. Supports wildcards.
metricsNeverDropList: # [array of strings] Metrics never drop list - List metric
names for which to ensure delivery. Supports wildcards.
metricsFieldsBlacklist: # [array of strings] Disable field metrics - List of event
fields for which to disable metric collection.
metricsDirectory: # [string] Metrics directory - Directory to store metrics on disk.
cpuProfileTTL: # [string] CPU profile TTL - The time-to-live for collected CPU
profiles.
eventsMetadataSources: # [array of strings] Event metadata sources - List of event
metadata sources to enable.
```

14.3.11. logger.yml

`logger.yml` maintains logging levels and redactions, per channel. In the UI, you can configure these at **Settings** > **Global Settings** > **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, warn, info, http, verbose, debug, silly
  DEFAULT: info
  input:DistMaster: debug
  output:DistWorker: debug
```

14.3.12. 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]
```

14.3.13. 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]
```

14.3.14. 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
will be used whenever a nonexistent/deleted output is referenced.
    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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
POST.
    format: # [string] Format - Specifies how to format events before sending out.
Defaults to NDJSON.

    # -------------- if format is custom ---------------

    customSourceExpression: # [string] Source expression - Expression to evaluate as
event. E.g., `${fieldA}, ${fieldB}`. Defaults to __httpOut (i.e. value of field
__httpOut).
    customDropWhenNull: # [boolean] Drop when null - Whether or not to drop events
when the source expression evaluates to null.
    customEventDelimiter: # [string] Event delimiter - Delimiter string to insert
between individual events. Defaults to newline character.
    customContentType: # [string] Content type - Content type to use for request.
Defaults to application/x-ndjson. Any content types set in Advanced Settings > Extra
HTTP headers will override this entry.

    # -------------------------------------------------------

    concurrency: # [number] Request concurrency - Maximum number of ongoing requests
before blocking.
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the
request body.
    maxPayloadEvents: # [number] Max events per request - Max number of events to
include in the request body. Default is 0 (unlimited).
    compress: # [boolean] Compress - Whether to compress the payload body before
sending.
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to No.
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
a request to complete before aborting it.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
Small values could cause the payload size to be smaller than the configured Max body
size.
    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
lookup. When a DNS server returns multiple addresses, this will cause Stream to
cycle through them in the order returned.
```

```
    failedRequestLoggingMode: # [string] Failed request logging mode - Determines
which data should be logged when a request fails. Defaults to None.  All headers are
redacted by default, except those listed under `Safe Headers`.
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe
to log in plain text.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ----------------------------------------------------------

    authType: # [string] Authentication type - The authentication method to use for
the HTTP request. Defaults to None.

    # -------------- 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 references your credentials

    # ----------------------------------------------------------


    # -------------- if authType is textSecret ---------------

    textSecret: # [string] Token (text secret) - Select (or create) a stored text
secret

    # ----------------------------------------------------------
```

```
    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  devnull_output: # [object]
    type: # [string] Output Type
    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  syslog_output: # [object]
    type: # [string] Output Type
    protocol: # [string] Protocol - The network protocol to use for sending out
syslog messages

    # -------------- if protocol is tcp ---------------

    loadBalanced: # [boolean] Load balancing - Use load-balanced destinations
    connectionTimeout: # [number] Connection Timeout - Amount of time (milliseconds)
to wait for the connection to establish before retrying
    writeTimeout: # [number] Write Timeout - Amount of time (milliseconds) to wait
for a write to complete before assuming connection is dead
    tls: # [object] TLS settings (client side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to No.
      servername: # [string] Server name (SNI) - Server name for the SNI (Server
Name Indication) TLS extension. It must be a host name, and not an IP address.
      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      caPath: # [string] CA certificate path - Path on client in which to find CA
certificates to verify the server's cert. PEM format. Can reference $ENV_VARS.
      privKeyPath: # [string] Private key path (mutual auth) - Path on client in
which to find the private key to use. PEM format. Can reference $ENV_VARS.
      certPath: # [string] Certificate path (mutual auth) - Path on client in which
to find certificates to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to use when
connecting
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when
connecting

      # ------------------------------------------------------

    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.
```

```
    # ------------------------------------------------------


    # -------------- 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
max record size is > than MTU then UDP packets can be fragmented across, set this
value  <= MTU to avoid fragmentation.

    # ------------------------------------------------------

    facility: # [number] Facility - Default value for message facility, will be
overwritten by value of __facility if set. Defaults to user.
    severity: # [number] Severity - Default value for message severity, will be
overwritten by value of __severity if set. Defaults to notice.
    appName: # [string] App Name - Default value for application name, will be
overwritten by value of __appname if set. Defaults to Cribl.
    messageFormat: # [string] Message Format - The syslog message format depending
on the receiver's support
    timestampFormat: # [string] Timestamp Format - Timestamp format to use when
serializing event's time field
    throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to
throttle while writing to an output. Also takes values with multiple-byte units,
such as KB, MB, GB, etc. (E.g., 42 MB.) Default value of 0 specifies no throttling.
    octetCountFraming: # [boolean] Octet count framing - When enabled, messages will
be prefixed with the byte count of the message. Otherwise, no prefix will be set,
and the message will be appended with a \n.
    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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
nested fields into index-time fields.
    throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to
throttle while writing to an output. Also takes values with multiple-byte units,
such as KB, MB, GB, etc. (E.g., 42 MB.) Default value of 0 specifies no throttling.
    connectionTimeout: # [number] Connection Timeout - Amount of time (milliseconds)
to wait for the connection to establish before retrying
    writeTimeout: # [number] Write Timeout - Amount of time (milliseconds) to wait
for a write to complete before assuming connection is dead
    tls: # [object] TLS settings (client side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to No.
```

```
      servername: # [string] Server name (SNI) - Server name for the SNI (Server
Name Indication) TLS extension. It must be a host name, and not an IP address.
      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      caPath: # [string] CA certificate path - Path on client in which to find CA
certificates to verify the server's cert. PEM format. Can reference $ENV_VARS.
      privKeyPath: # [string] Private key path (mutual auth) - Path on client in
which to find the private key to use. PEM format. Can reference $ENV_VARS.
      certPath: # [string] Certificate path (mutual auth) - Path on client in which
to find certificates to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to use when
connecting
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when
connecting

      # ----------------------------------------------------------

    enableMultiMetrics: # [boolean] Output multiple metrics - Output metrics in
multiple-metric format in a single event. Supported in Splunk 8.0 and above.
    enableACK: # [boolean] Minimize in-flight data loss - Check if indexer is
shutting down and stop sending data. This helps minimize data loss during shutdown.
    maxS2Sversion: # [string] Max S2S version - The highest S2S protocol version to
advertise during handshake.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ----------------------------------------------------------

    authType: # [string] Authentication method - Enter a token directly, or provide
a secret referencing a token
    authToken: # [string] Auth token - Shared secret token to use when establishing
a connection to a Splunk indexer.

    # -------------- if authType is manual ---------------


    # ----------------------------------------------------------

    textSecret: # [string] Auth token (text secret) - Select (or create) a stored
text secret

    # -------------- if authType is secret ---------------
```

```
    # ------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  splunk_lb_output: # [object]
    type: # [string] Output Type
    dnsResolvePeriodSec: # [number] DNS resolution period (seconds) - Re-resolve any
hostnames every this many seconds and pick up destinations from A records.
    loadBalanceStatsPeriodSec: # [number] Load balance stats period (seconds) - How
far back in time to keep traffic stats for load balancing purposes.
    maxConcurrentSenders: # [number] Max connections - Maximum number of concurrent
connections (per worker process). A random set of IPs will be picked on every DNS
resolution period. Use 0 for unlimited.
    nestedFields: # [string] Nested field serialization - Specifies how to serialize
nested fields into index-time fields.
    throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to
throttle while writing to an output. Also takes values with multiple-byte units,
such as KB, MB, GB, etc. (E.g., 42 MB.) Default value of 0 specifies no throttling.
    connectionTimeout: # [number] Connection Timeout - Amount of time (milliseconds)
to wait for the connection to establish before retrying
    writeTimeout: # [number] Write Timeout - Amount of time (milliseconds) to wait
for a write to complete before assuming connection is dead
    tls: # [object] TLS settings (client side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to No.
      servername: # [string] Server name (SNI) - Server name for the SNI (Server
Name Indication) TLS extension. It must be a host name, and not an IP address.
      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      caPath: # [string] CA certificate path - Path on client in which to find CA
certificates to verify the server's cert. PEM format. Can reference $ENV_VARS.
      privKeyPath: # [string] Private key path (mutual auth) - Path on client in
which to find the private key to use. PEM format. Can reference $ENV_VARS.
      certPath: # [string] Certificate path (mutual auth) - Path on client in which
to find certificates to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to use when
connecting
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when
connecting

      # -------------------------------------------------------

    enableMultiMetrics: # [boolean] Output multiple metrics - Output metrics in
multiple-metric format in a single event. Supported in Splunk 8.0 and above.
    enableACK: # [boolean] Minimize in-flight data loss - Check if indexer is
```

```
shutting down and stop sending data. This helps minimize data loss during shutdown.
    maxS2Sversion: # [string] Max S2S version - The highest S2S protocol version to
advertise during handshake.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ----------------------------------------------------------

    indexerDiscovery: # [boolean] Indexer Discovery - Automatically discover
indexers in indexer clustering environment.

    # -------------- if indexerDiscovery is true ---------------

    indexerDiscoveryConfigs: # [object]  - List of configurations to set up indexer
discovery in Splunk Indexer clustering environment.
      site: # [string] [required] Site - Clustering site of the indexers from where
indexers need to be discovered. In case of single site cluster, it defaults to
'default' site.
      masterUri: # [string] Cluster Manager URI - Full URI of Splunk cluster Manager
(scheme://host:port). E.g.: https://managerAddress:8089
      refreshIntervalSec: # [number] [required] Refresh Period - Time interval in
seconds between two consecutive indexer list fetches from cluster Manager.
      authType: # [string] Authentication method - Enter a token directly, or
provide a secret referencing a token
      authToken: # [string] Auth token - Authentication token required to
authenticate to cluster Manager for indexer discovery.

      # -------------- if authType is manual ---------------


      # ----------------------------------------------------------

      textSecret: # [string] Auth token (text secret) - Select (or create) a stored
text secret

      # -------------- if authType is secret ---------------


      # ----------------------------------------------------------


    # ----------------------------------------------------------
```

```
    # -------------- if indexerDiscovery is false ---------------

    excludeSelf: # [boolean] Exclude current host IPs - Exclude all IPs of the
current host from the list of any resolved hostnames.
    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
disable TLS.
        servername: # [string] TLS Servername - Servername to use if establishing a
TLS connection. If not specified, defaults to connection host (iff not an IP);
otherwise, to the global TLS settings.
        weight: # [number] Load Weight - The weight to use for load-balancing
purposes.
    dnsResolvePeriodSec: # [number] DNS resolution period (seconds) - Re-resolve any
hostnames every this many seconds and pick up destinations from A records.
    loadBalanceStatsPeriodSec: # [number] Load balance stats period (seconds) - How
far back in time to keep traffic stats for load balancing purposes.
    maxConcurrentSenders: # [number] Max connections - Maximum number of concurrent
connections (per worker process). A random set of IPs will be picked on every DNS
resolution period. Use 0 for unlimited.

    # ------------------------------------------------------

    authType: # [string] Authentication method - Enter a token directly, or provide
a secret referencing a token
    authToken: # [string] Auth token - Shared secret token to use when establishing
a connection to a Splunk indexer.

    # -------------- if authType is manual ---------------


    # ------------------------------------------------------

    textSecret: # [string] Auth token (text secret) - Select (or create) a stored
text secret

    # -------------- if authType is secret ---------------


    # ------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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 to, e.g., http://localhost:8088/services/collector/event
```

```
    useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS
lookup. When a DNS server returns multiple addresses, this will cause Stream to
cycle through them in the order returned.

    # -------------------------------------------------------


    # -------------- if loadBalanced is true ---------------

    excludeSelf: # [boolean] Exclude current host IPs - Exclude all IPs of the
current host from the list of any resolved hostnames.
    urls: # [array] Splunk HEC Endpoints
      - url: # [string] HEC Endpoint - URL to a Splunk HEC endpoint to send events
to, e.g., http://localhost:8088/services/collector/event
        weight: # [number] Load Weight - The weight to use for load-balancing
purposes.
    dnsResolvePeriodSec: # [number] DNS resolution period (seconds) - Re-resolve any
hostnames every this many seconds and pick up destinations from A records.
    loadBalanceStatsPeriodSec: # [number] Load balance stats period (seconds) - How
far back in time to keep traffic stats for load balancing purposes.

    # -------------------------------------------------------

    nextQueue: # [string] Next Processing Queue - Which Splunk processing queue to
send the events after HEC processing
    tcpRouting: # [string] Default _TCP_ROUTING - Set the value of _TCP_ROUTING for
events that does not have _ctrl._TCP_ROUTING set
    concurrency: # [number] Request concurrency - Maximum number of ongoing requests
before blocking.
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the
request body.
    maxPayloadEvents: # [number] Max events per request - Max number of events to
include in the request body. Default is 0 (unlimited).
    compress: # [boolean] Compress - Whether to compress the payload body before
sending.
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to No.
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
a request to complete before aborting it.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
Small values could cause the payload size to be smaller than the configured Max body
size.
    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
which data should be logged when a request fails. Defaults to None.  All headers are
redacted by default, except those listed under `Safe Headers`.
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe
to log in plain text.
    enableMultiMetrics: # [boolean] Output multi-metrics - Output metrics in
multiple-metric format, supported in Splunk 8.0 and above to allow multiple metrics
in a single event.
    authType: # [string] Authentication method - Enter a token directly, or provide
a secret referencing a token

    # -------------- if authType is manual ---------------

    token: # [string] HEC Auth token - Splunk HEC authentication token
```

```
    # ------------------------------------------------------


    # -------------- if authType is secret ---------------

    textSecret: # [string] HEC Auth token (text secret) - Select (or create) a
stored text secret

    # ------------------------------------------------------

    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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
current host from the list of any resolved hostnames.
    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
disable TLS.
        servername: # [string] TLS Servername - Servername to use if establishing a
TLS connection. If not specified, defaults to connection host (iff not an IP);
otherwise, to the global TLS settings.
        weight: # [number] Load Weight - The weight to use for load-balancing
purposes.
    dnsResolvePeriodSec: # [number] DNS resolution period (seconds) - Re-resolve any
hostnames every this many seconds and pick up destinations from A records.
    loadBalanceStatsPeriodSec: # [number] Load balance stats period (seconds) - How
far back in time to keep traffic stats for load balancing purposes.
    maxConcurrentSenders: # [number] Max connections - Maximum number of concurrent
connections (per worker process). A random set of IPs will be picked on every DNS
resolution period. Use 0 for unlimited.

    # ------------------------------------------------------

    compression: # [string] Compression - Codec to use to compress the data before
sending
    throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to
throttle while writing to an output. Also takes values with multiple-byte units,
such as KB, MB, GB, etc. (E.g., 42 MB.) Default value of 0 specifies no throttling.
    tls: # [object] TLS settings (client side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to No.
      servername: # [string] Server name (SNI) - Server name for the SNI (Server
Name Indication) TLS extension. It must be a host name, and not an IP address.
      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      caPath: # [string] CA certificate path - Path on client in which to find CA
certificates to verify the server's cert. PEM format. Can reference $ENV_VARS.
      privKeyPath: # [string] Private key path (mutual auth) - Path on client in
which to find the private key to use. PEM format. Can reference $ENV_VARS.
      certPath: # [string] Certificate path (mutual auth) - Path on client in which
to find certificates to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to use when
connecting
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when
connecting

      # ------------------------------------------------------

    connectionTimeout: # [number] Connection Timeout - Amount of time (milliseconds)
to wait for the connection to establish before retrying
    writeTimeout: # [number] Write Timeout - Amount of time (milliseconds) to wait
for a write to complete before assuming connection is dead
    tokenTTLMinutes: # [number] Auth Token TTL minutes - The number of minutes
before the internally generated authentication token expires, valid values between 1
and 60
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------
```

```
    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ------------------------------------------------------

    authType: # [string] Authentication method - Enter a token directly, or provide
a secret referencing a token
    authToken: # [string] Auth token - Optional authentication token to include as
part of the connection header

    # -------------- if authType is manual ---------------


    # ------------------------------------------------------

    textSecret: # [string] Auth token (text secret) - Select (or create) a stored
text secret

    # -------------- if authType is secret ---------------


    # ------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  wavefront_output: # [object]
    type: # [string] Output Type
    authType: # [string] Authentication method - Enter a token directly, or provide
a secret referencing a token

    # -------------- if authType is manual ---------------

    token: # [string] Auth token - WaveFront API authentication token (see [here]
(https://docs.wavefront.com/wavefront_api.html#generating-an-api-token))

    # ------------------------------------------------------


    # -------------- if authType is secret ---------------
```

```
    textSecret: # [string] Auth token (text secret) - Select (or create) a stored
text secret

    # ------------------------------------------------------

    domain: # [string] Domain name - WaveFront domain name, e.g. "longboard"
    concurrency: # [number] Request concurrency - Maximum number of ongoing requests
before blocking.
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the
request body.
    maxPayloadEvents: # [number] Max events per request - Max number of events to
include in the request body. Default is 0 (unlimited).
    compress: # [boolean] Compress - Whether to compress the payload body before
sending.
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to Yes.
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
a request to complete before aborting it.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
Small values could cause the payload size to be smaller than the configured Max body
size.
    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
lookup. When a DNS server returns multiple addresses, this will cause Stream to
cycle through them in the order returned.
    failedRequestLoggingMode: # [string] Failed request logging mode - Determines
which data should be logged when a request fails. Defaults to None.  All headers are
redacted by default, except those listed under `Safe Headers`.
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe
to log in plain text.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
```

```
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  signalfx_output: # [object]
    type: # [string] Output Type
    authType: # [string] Authentication method - Enter a token directly, or provide
a secret referencing a token

    # -------------- if authType is manual ---------------

    token: # [string] Auth token - SignalFx API access token (see [here]
(https://docs.signalfx.com/en/latest/admin-guide/tokens.html#working-with-access-
tokens))

    # -----------------------------------------------------


    # -------------- if authType is secret ---------------

    textSecret: # [string] Auth token (text secret) - Select (or create) a stored
text secret

    # -----------------------------------------------------

    realm: # [string] Realm - SignalFx realm name, e.g. "us0"
    concurrency: # [number] Request concurrency - Maximum number of ongoing requests
before blocking.
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the
request body.
    maxPayloadEvents: # [number] Max events per request - Max number of events to
include in the request body. Default is 0 (unlimited).
    compress: # [boolean] Compress - Whether to compress the payload body before
sending.
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to Yes.
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
a request to complete before aborting it.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
Small values could cause the payload size to be smaller than the configured Max body
size.
    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
lookup. When a DNS server returns multiple addresses, this will cause Stream to
cycle through them in the order returned.
    failedRequestLoggingMode: # [string] Failed request logging mode - Determines
which data should be logged when a request fails. Defaults to None.  All headers are
redacted by default, except those listed under `Safe Headers`.
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe
to log in plain text.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
```

```
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # --------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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
files before compressing and moving to final destination. Use performant stable
storage.
    addIdToStagePath: # [boolean] Add output ID - Append output's ID to staging
location.
    removeEmptyDirs: # [boolean] Remove staging dirs - Remove empty staging
directories after moving files.

    # -------------- if removeEmptyDirs is true ---------------

    emptyDirCleanupSec: # [number] Staging cleanup period - How often (secs) to
clean-up empty directories when 'Remove Staging Dirs' is enabled.

    # --------------------------------------------------------

    partitionExpr: # [string] Partitioning expression - JS expression defining how
files are partitioned and organized. Default is date-based. If blank, Stream will
fall back to the event's __partition field value â if present â otherwise to
each location's root directory.
    format: # [string] Data format - Format of the output data.

    # -------------- if format is json ---------------

    compress: # [string] Compress - Choose data compression format to apply before
moving files to final destination.

    # --------------------------------------------------------


    # -------------- if format is parquet ---------------

    parquetSchema: # [string] Parquet schema - Select a Parquet schema. New schemas
can be uploaded under Processing > Knowledge > Parquet Schemas
```

```
    parquetRowGroupSize: # [string] Row group size - Ideal memory size for row group
segments. E.g., 128MB or 1GB. Affects memory use when writing. Imposes a target, not
a strict limit; row groups final size may be larger or smaller.
    parquetPageSize: # [string] Page size - Ideal memory size for page segments.
E.g., 1MB or 128MB. Generally, lower values improve reading speed, while higher
values improve compression. Imposes a target, not a strict limit; pages final size
may be larger or smaller.
    spacer: # [null]
    parquetVersion: # [string] Parquet version - Determines which data types are
supported and how they are represented.
    parquetDataPageVersion: # [string] Data page version - Serialization format of
data pages. Note that not all reader implemtations support Data page V2.
    shouldLogInvalidRows: # [boolean] Log invalid rows - Output up to 20 unique rows
that were skipped due to data format mismatch. Must have Logging set to Debug to see
output.

    # ------------------------------------------------------

    baseFileName: # [string] File name prefix expression - JavaScript expression to
define the output filename prefix (can be constant).
    fileNameSuffix: # [string] File name suffix expression - JavaScript expression
to define the output filename suffix (can be constant).  The `__format` variable
refers to the value of the `Data format` field (`json` or `raw`).  The
`__compression` field refers to the kind of compression being used (`none` or
`gzip`)
    maxFileSizeMB: # [number] Max file size (MB) - Maximum uncompressed output file
size. Files of this size will be closed and moved to final output location.
    maxFileOpenTimeSec: # [number] 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.
    maxFileIdleTimeSec: # [number] 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.
    maxOpenFiles: # [number] Max open files - Maximum number of files to keep open
concurrently. When over, the oldest open files will be closed and moved to final
output location.
    onBackpressure: # [string] Backpressure behavior - Whether to block or drop
events when all receivers are exerting backpressure.
    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  s3_output: # [object]
    type: # [string] Output Type
    bucket: # [string] S3 bucket name - Name of the destination S3 bucket. Must be a
JavaScript expression (which can evaluate to a constant value), enclosed in quotes
or backticks. Can be evaluated only at init time. E.g., referencing a Global
Variable: `myBucket-${C.vars.myVar}`.
    region: # [string] Region - Region where the S3 bucket is located.
    awsAuthenticationMethod: # [string] Authentication method - AWS authentication
method. Choose Auto to use IAM roles.

    # -------------- 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
references your access key and secret key.

    # -------------------------------------------------------

    awsSecretKey: # [string] Secret key - Secret key
    endpoint: # [string] Endpoint - S3 service endpoint. If empty, defaults to AWS'
Region-specific endpoint. Otherwise, it must point to S3-compatible endpoint.
    signatureVersion: # [string] Signature version - Signature version to use for
signing S3 requests.
    reuseConnections: # [boolean] Reuse connections - Whether to reuse connections
between requests, which can improve performance.
    rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to
reject certificates that cannot be verified against a valid CA (e.g., self-signed
certificates).
    enableAssumeRole: # [boolean] Enable for S3 - Use Assume Role credentials to
access S3
    assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the
role to assume
    assumeRoleExternalId: # [string] External ID - External ID to use when assuming
role
    stagePath: # [string] [required] Staging location - Filesystem location in which
to buffer files, before compressing and moving to final destination. Use performant
stable storage.
    addIdToStagePath: # [boolean] Add output ID - Append output's ID to staging
location.
    removeEmptyDirs: # [boolean] Remove staging dirs - Remove empty staging
directories after moving files.

    # -------------- if removeEmptyDirs is true ---------------

    emptyDirCleanupSec: # [number] Staging cleanup period - How often (secs) to
clean-up empty directories when 'Remove Staging Dirs' is enabled.

    # -------------------------------------------------------

    destPath: # [string] [required] Key prefix - Prefix to append to files before
uploading. Must be a JavaScript expression (which can evaluate to a constant value),
enclosed in quotes or backticks. Can be evaluated only at init time. E.g.,
referencing a Global Variable: `myKeyPrefix-${C.vars.myVar}`.
    objectACL: # [string] Object ACL - Object ACL to assign to uploaded objects.
    storageClass: # [string] Storage class - Storage class to select for uploaded
objects.
    serverSideEncryption: # [string] Server side encryption - Server-side encryption
for uploaded objects.

    # -------------- if serverSideEncryption is aws:kms ---------------

    kmsKeyId: # [string] KMS key ID - ID or ARN of the KMS customer managed key to
use for encryption

    # -------------------------------------------------------

    partitionExpr: # [string] Partitioning expression - JS expression defining how
```

```
files are partitioned and organized. Default is date-based. If blank, Stream will
fall back to the event's __partition field value â if present â otherwise to
each location's root directory.
    format: # [string] Data format - Format of the output data.

    # -------------- if format is json ---------------

    compress: # [string] Compress - Choose data compression format to apply before
moving files to final destination.

    # --------------------------------------------------------


    # -------------- if format is parquet ---------------

    parquetSchema: # [string] Parquet schema - Select a Parquet schema. New schemas
can be uploaded under Processing > Knowledge > Parquet Schemas
    parquetRowGroupSize: # [string] Row group size - Ideal memory size for row group
segments. E.g., 128MB or 1GB. Affects memory use when writing. Imposes a target, not
a strict limit; row groups final size may be larger or smaller.
    parquetPageSize: # [string] Page size - Ideal memory size for page segments.
E.g., 1MB or 128MB. Generally, lower values improve reading speed, while higher
values improve compression. Imposes a target, not a strict limit; pages final size
may be larger or smaller.
    spacer: # [null]
    parquetVersion: # [string] Parquet version - Determines which data types are
supported and how they are represented.
    parquetDataPageVersion: # [string] Data page version - Serialization format of
data pages. Note that not all reader implemtations support Data page V2.
    shouldLogInvalidRows: # [boolean] Log invalid rows - Output up to 20 unique rows
that were skipped due to data format mismatch. Must have Logging set to Debug to see
output.

    # --------------------------------------------------------

    baseFileName: # [string] File name prefix expression - JavaScript expression to
define the output filename prefix (can be constant).
    fileNameSuffix: # [string] File name suffix expression - JavaScript expression
to define the output filename suffix (can be constant).  The `__format` variable
refers to the value of the `Data format` field (`json` or `raw`).  The
`__compression` field refers to the kind of compression being used (`none` or
`gzip`)
    maxFileSizeMB: # [number] Max file size (MB) - Maximum uncompressed output file
size. Files of this size will be closed and moved to final output location.
    maxFileOpenTimeSec: # [number] 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.
    maxFileIdleTimeSec: # [number] 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.
    maxOpenFiles: # [number] Max open files - Maximum number of files to keep open
concurrently. When over, the oldest open files will be closed and moved to final
output location.
    onBackpressure: # [string] Backpressure behavior - Whether to block or drop
events when all receivers are exerting backpressure.
    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
```

```
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  azure_blob_output: # [object]
    type: # [string] Output Type
    containerName: # [string] Container name - A container organizes a set of blobs,
similar to a directory in a file system.
    createContainer: # [boolean] Create container - Creates the configured container
in Azure Blob Storage if it does not already exist.
    destPath: # [string] [required] Blob prefix - Root directory prepended to path
before uploading.
    stagePath: # [string] [required] Staging location - Filesystem location in which
to buffer files, before compressing and moving to final destination. Use performant
stable storage.
    addIdToStagePath: # [boolean] Add output ID - Append output's ID to staging
location.
    removeEmptyDirs: # [boolean] Remove staging dirs - Remove empty staging
directories after moving files.

    # -------------- if removeEmptyDirs is true ---------------

    emptyDirCleanupSec: # [number] Staging cleanup period - How often (secs) to
clean-up empty directories when 'Remove Staging Dirs' is enabled.

    # ----------------------------------------------------------

    partitionExpr: # [string] Partitioning expression - JS expression defining how
files are partitioned and organized. Default is date-based. If blank, Stream will
fall back to the event's __partition field value â if present â otherwise to
each location's root directory.
    format: # [string] Data format - Format of the output data.

    # -------------- if format is json ---------------

    compress: # [string] Compress - Choose data compression format to apply before
moving files to final destination.

    # ----------------------------------------------------------


    # -------------- if format is parquet ---------------

    parquetSchema: # [string] Parquet schema - Select a Parquet schema. New schemas
can be uploaded under Processing > Knowledge > Parquet Schemas
    parquetRowGroupSize: # [string] Row group size - Ideal memory size for row group
segments. E.g., 128MB or 1GB. Affects memory use when writing. Imposes a target, not
a strict limit; row groups final size may be larger or smaller.
    parquetPageSize: # [string] Page size - Ideal memory size for page segments.
E.g., 1MB or 128MB. Generally, lower values improve reading speed, while higher
values improve compression. Imposes a target, not a strict limit; pages final size
may be larger or smaller.
    spacer: # [null]
    parquetVersion: # [string] Parquet version - Determines which data types are
supported and how they are represented.
    parquetDataPageVersion: # [string] Data page version - Serialization format of
data pages. Note that not all reader implemtations support Data page V2.
    shouldLogInvalidRows: # [boolean] Log invalid rows - Output up to 20 unique rows
that were skipped due to data format mismatch. Must have Logging set to Debug to see
output.
```

```
    # ------------------------------------------------------

    baseFileName: # [string] File name prefix expression - JavaScript expression to
define the output filename prefix (can be constant).
    fileNameSuffix: # [string] File name suffix expression - JavaScript expression
to define the output filename suffix (can be constant).  The `__format` variable
refers to the value of the `Data format` field (`json` or `raw`).  The
`__compression` field refers to the kind of compression being used (`none` or
`gzip`)
    maxFileSizeMB: # [number] Max file size (MB) - Maximum uncompressed output file
size. Files of this size will be closed and moved to final output location.
    maxFileOpenTimeSec: # [number] 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.
    maxFileIdleTimeSec: # [number] 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.
    maxOpenFiles: # [number] Max open files - Maximum number of files to keep open
concurrently. When over, the oldest open files will be closed and moved to final
output location.
    onBackpressure: # [string] Backpressure behavior - Whether to block or drop
events when all receivers are exerting backpressure.
    authType: # [string] Authentication method - Enter connection string directly,
or select a stored secret
    connectionString: # [string] Connection string - Enter your Azure Storage
account connection string. If left blank, Stream will fall back to
env.AZURE_STORAGE_CONNECTION_STRING.

    # -------------- if authType is manual ---------------


    # ------------------------------------------------------

    textSecret: # [string] Connection string (text secret) - Select (or create) a
stored text secret

    # -------------- if authType is secret ---------------


    # ------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  azure_logs_output: # [object]
    type: # [string] Output Type
    logType: # [string] Log Type - The Log Type of events sent to this LogAnalytics
workspace. Can be overwritten by event field __logType.
    resourceId: # [string] Resource ID - Optional Resource ID of the Azure resource
the data should be associated with, can be overridden by event field __resourceId.
This populates the _ResourceId property and allows the data to be included in
resource-centric queries. If this field isn't specified, the data will not be
included in resource-centric queries.
```

```
    concurrency: # [number] Request concurrency - Maximum number of ongoing requests
before blocking.
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the
request body.
    maxPayloadEvents: # [number] Max events per request - Max number of events to
include in the request body. Default is 0 (unlimited).
    compress: # [boolean] Compress - Whether to compress the payload body before
sending.
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to Yes.
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
a request to complete before aborting it.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
Small values could cause the payload size to be smaller than the configured Max body
size.
    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
lookup. When a DNS server returns multiple addresses, this will cause Stream to
cycle through them in the order returned.
    failedRequestLoggingMode: # [string] Failed request logging mode - Determines
which data should be logged when a request fails. Defaults to None.  All headers are
redacted by default, except those listed under `Safe Headers`.
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe
to log in plain text.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ----------------------------------------------------------

    authType: # [string] Authentication method - Enter workspace ID and workspace
key directly, or select a stored secret
    workspaceId: # [string] Workspace ID - Azure Log Analytics Workspace ID. See
Azure Dashboard WorkspaceÂ > Advanced settings.
    workspaceKey: # [string] Workspace key - Azure Log Analytics Workspace Primary
or Secondary Shared Key. See Azure Dashboard WorkspaceÂ > Advanced settings.

    # -------------- if authType is manual ---------------


    # ----------------------------------------------------------
```

```
    keypairSecret: # [string] Secret key pair - Select (or create) a stored secret
that references your access key and secret key.

    # -------------- if authType is secret ---------------


    # -----------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  kinesis_output: # [object]
    type: # [string] Output Type
    streamName: # [string] Stream Name - Kinesis stream name to send events to.
    awsAuthenticationMethod: # [string] Authentication method - AWS authentication
method. Choose Auto to use IAM roles.

    # -------------- 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
references your access key and secret key.

    # -----------------------------------------------------

    awsSecretKey: # [string] Secret key - Secret key
    region: # [string] [required] Region - Region where the Kinesis stream is
located
    endpoint: # [string] Endpoint - Kinesis stream service endpoint. If empty,
defaults to AWS' Region-specific endpoint. Otherwise, it must point to Kinesis
stream-compatible endpoint.
    signatureVersion: # [string] Signature version - Signature version to use for
signing Kinesis stream requests.
    reuseConnections: # [boolean] Reuse connections - Whether to reuse connections
between requests, which can improve performance.
    rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to
reject certificates that cannot be verified against a valid CA (e.g., self-signed
certificates).
    enableAssumeRole: # [boolean] Enable for Kinesis stream - Use Assume Role
credentials to access Kinesis stream
    assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the
role to assume
    assumeRoleExternalId: # [string] External ID - External ID to use when assuming
role
    concurrency: # [number] Put request concurrency - Maximum number of ongoing put
requests before blocking.
    maxRecordSizeKB: # [number] Max record size (KB, uncompressed) - Maximum size
```

```
(KB) of each individual record before compression. For non-compressible data 1MB is
the max recommended size
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
Small values could cause the payload size to be smaller than the configured Max
record size.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ----------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  honeycomb_output: # [object]
    type: # [string] Output Type
    dataset: # [string] Dataset name - Name of the dataset to send events to â
e.g., observability
    concurrency: # [number] Request concurrency - Maximum number of ongoing requests
before blocking.
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the
request body.
    maxPayloadEvents: # [number] Max events per request - Max number of events to
include in the request body. Default is 0 (unlimited).
    compress: # [boolean] Compress - Whether to compress the payload body before
sending.
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to Yes.
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
a request to complete before aborting it.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
Small values could cause the payload size to be smaller than the configured Max body
size.
    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
```

```
lookup. When a DNS server returns multiple addresses, this will cause Stream to
cycle through them in the order returned.
    failedRequestLoggingMode: # [string] Failed request logging mode - Determines
which data should be logged when a request fails. Defaults to None.  All headers are
redacted by default, except those listed under `Safe Headers`.
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe
to log in plain text.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # --------------------------------------------------------

    authType: # [string] Authentication method - Enter API key directly, or select a
stored secret
    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
secret

    # -------------- if authType is secret ---------------


    # --------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  azure_eventhub_output: # [object]
    type: # [string] Output Type
    brokers: # [array of strings] Brokers - List of Event Hubs Kafka brokers to
connect to, eg. yourdomain.servicebus.windows.net:9093. The hostname can be found in
the host portion of the primary or secondary connection string in Shared Access
```

```
Policies.
    topic: # [string] [required] Event Hub Name - The name of the Event Hub (a.k.a.
Kafka Topic) to publish events. Can be overwritten using field __topicOut.
    ack: # [number] Acknowledgments - Control the number of required acknowledgments
    format: # [string] Record data format - Format to use to serialize events before
writing to the Event Hubs Kafka brokers.
    maxRecordSizeKB: # [number] Max record size (KB, uncompressed) - Maximum size
(KB) of each record batch before compression. Setting should be < message.max.bytes
settings in Event Hubs brokers.
    flushEventCount: # [number] Max events per batch - Maximum number of events in a
batch before forcing a flush.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
Small values could cause the payload size to be smaller than the configured Max
record size.
    connectionTimeout: # [number] Connection timeout (ms) - Maximum time to wait for
a successful connection.
    requestTimeout: # [number] Request timeout (ms) - Maximum time to wait for a
successful request.
    sasl: # [object] Authentication - Authentication parameters to use when
connecting to brokers. Using TLS is highly recommended.
      disabled: # [boolean] Disabled - Enable authentication.

      # -------------- if disabled is false ---------------

      mechanism: # [string] SASL mechanism - SASL authentication mechanism to use.
PLAIN is the only mechanism currently supported for Event Hubs Kafka brokers.
      username: # [string] Username - The username for authentication. For Event
Hubs, this should always be $ConnectionString.
      authType: # [string] Authentication method - Enter password directly, or
select a stored secret

      # ---------------------------------------------------------

    tls: # [object] TLS settings (client side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      rejectUnauthorized: # [boolean] Validate server certs - For Event Hubs, this
should always be false.

      # ---------------------------------------------------------

    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
```

```
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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 secret

    # -------------------------------------------------------

    logType: # [string] Log type - Log type value to send to Chronicle. Can be
overwritten by event field __logType.
    logTextField: # [string] Log text field - Name of the event field that contains
the log text to send. If not specified, Stream sends a JSON representation of the
whole event.
    logFormatType: # [string] Send events as
    region: # [string] Region - Regional endpoint to send events to
    concurrency: # [number] Request concurrency - Maximum number of ongoing requests
before blocking.
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the
request body.
    maxPayloadEvents: # [number] Max events per request - Max number of events to
include in the request body. Default is 0 (unlimited).
    compress: # [boolean] Compress - Whether to compress the payload body before
sending.
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to Yes.
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
a request to complete before aborting it.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
Small values could cause the payload size to be smaller than the configured Max body
size.
    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
lookup. When a DNS server returns multiple addresses, this will cause Stream to
```

```
cycle through them in the order returned.
    failedRequestLoggingMode: # [string] Failed request logging mode - Determines
which data should be logged when a request fails. Defaults to None.  All headers are
redacted by default, except those listed under `Safe Headers`.
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe
to log in plain text.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ----------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  google_cloud_storage_output: # [object]
    type: # [string] Output Type
    bucket: # [string] Bucket name - Name of the destination Bucket. This value can
be a constant or a JavaScript expression that can only be evaluated at init time.
E.g. referencing a Global Variable: `myBucket-${C.vars.myVar}`.
    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
signing Google Cloud Storage requests.
    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
```

```
references your access key and secret key.

    # ------------------------------------------------------

    stagePath: # [string] [required] Staging location - Filesystem location in which
to buffer files, before compressing and moving to final destination. Use performant
stable storage.
    destPath: # [string] [required] Key prefix - Prefix to append to files before
uploading. Must be a JavaScript expression (which can evaluate to a constant value),
enclosed in quotes or backticks. Can be evaluated only at init time. E.g.,
referencing a Global Variable: `myKeyPrefix-${C.vars.myVar}`.
    objectACL: # [string] Object ACL - Object ACL to assign to uploaded objects.
    storageClass: # [string] Storage class - Storage class to select for uploaded
objects.
    reuseConnections: # [boolean] Reuse connections - Whether to reuse connections
between requests, which can improve performance.
    rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to
reject certificates that cannot be verified against a valid CA (e.g., self-signed
certificates).
    addIdToStagePath: # [boolean] Add output ID - Append output's ID to staging
location.
    removeEmptyDirs: # [boolean] Remove staging dirs - Remove empty staging
directories after moving files.

    # -------------- if removeEmptyDirs is true ---------------

    emptyDirCleanupSec: # [number] Staging cleanup period - How often (secs) to
clean-up empty directories when 'Remove Staging Dirs' is enabled.

    # -------------------------------------------------------

    partitionExpr: # [string] Partitioning expression - JS expression defining how
files are partitioned and organized. Default is date-based. If blank, Stream will
fall back to the event's __partition field value â if present â otherwise to
each location's root directory.
    format: # [string] Data format - Format of the output data.

    # -------------- if format is json ---------------

    compress: # [string] Compress - Choose data compression format to apply before
moving files to final destination.

    # -------------------------------------------------------


    # -------------- if format is parquet ---------------

    parquetSchema: # [string] Parquet schema - Select a Parquet schema. New schemas
can be uploaded under Processing > Knowledge > Parquet Schemas
    parquetRowGroupSize: # [string] Row group size - Ideal memory size for row group
segments. E.g., 128MB or 1GB. Affects memory use when writing. Imposes a target, not
a strict limit; row groups final size may be larger or smaller.
    parquetPageSize: # [string] Page size - Ideal memory size for page segments.
E.g., 1MB or 128MB. Generally, lower values improve reading speed, while higher
values improve compression. Imposes a target, not a strict limit; pages final size
may be larger or smaller.
    spacer: # [null]
    parquetVersion: # [string] Parquet version - Determines which data types are
supported and how they are represented.
    parquetDataPageVersion: # [string] Data page version - Serialization format of
```

```
data pages. Note that not all reader implemtations support Data page V2.
    shouldLogInvalidRows: # [boolean] Log invalid rows - Output up to 20 unique rows
that were skipped due to data format mismatch. Must have Logging set to Debug to see
output.

    # ------------------------------------------------------

    baseFileName: # [string] File name prefix expression - JavaScript expression to
define the output filename prefix (can be constant).
    fileNameSuffix: # [string] File name suffix expression - JavaScript expression
to define the output filename suffix (can be constant).  The `__format` variable
refers to the value of the `Data format` field (`json` or `raw`).  The
`__compression` field refers to the kind of compression being used (`none` or
`gzip`)
    maxFileSizeMB: # [number] Max file size (MB) - Maximum uncompressed output file
size. Files of this size will be closed and moved to final output location.
    maxFileOpenTimeSec: # [number] 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.
    maxFileIdleTimeSec: # [number] 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.
    maxOpenFiles: # [number] Max open files - Maximum number of files to keep open
concurrently. When over, the oldest open files will be closed and moved to final
output location.
    onBackpressure: # [string] Backpressure behavior - Whether to block or drop
events when all receivers are exerting backpressure.
    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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
exist.
    orderedDelivery: # [boolean] Ordered delivery - If enabled, send events in the
order they were added to the queue. For this to work correctly, the process
receiving events must have ordering enabled.
    region: # [string] Region - Region to publish messages to. Select 'default' to
allow Google to auto-select the nearest region. When using ordered delivery, the
selected region must be allowed by message storage policy.
    googleAuthMethod: # [string] Authentication Method - Google authentication
method. Choose Auto to use environment variables PUBSUB_PROJECT and
PUBSUB_CREDENTIALS.

    # -------------- if googleAuthMethod is manual ---------------

    serviceAccountCredentials: # [string] Service account credentials - Contents of
service account credentials (JSON keys) file downloaded from Google Cloud. To upload
a file, click the upload button at this field's upper right. As an alternative, you
can use environment variables (see [here](https://googleapis.dev/ruby/google-cloud-
pubsub/latest/file.AUTHENTICATION.html)).

    # ------------------------------------------------------
```

```
    # -------------- if googleAuthMethod is secret ---------------

    secret: # [string] Service account credentials (text secret) - Select (or
create) a stored text secret

    # ------------------------------------------------------

    batchSize: # [number] Batch size - The maximum number of items the Google API
should batch before it sends them to the topic.
    batchTimeout: # [number] Batch timeout (ms) - The maximum amount of time, in
milliseconds, that the Google API should wait to send a batch (if the Batch size is
not reached).
    maxQueueSize: # [number] Max queue size - Maximum number of queued batches
before blocking.
    maxRecordSizeKB: # [number] Max batch size (KB) - Maximum size (KB) of batches
to send.
    maxInProgress: # [number] Max concurrent requests - The maximum number of in-
progress API requests before backpressure is applied.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  kafka_output: # [object]
    type: # [string] Output Type
    brokers: # [array of strings] Brokers - List of Kafka brokers to connect to,
e.g., kafkaBrokerHost:9092.
    topic: # [string] [required] Topic - The topic to publish events to. Can be
overridden using the __topicOut field.
    ack: # [number] Acknowledgments - Control the number of required
acknowledgments.
    format: # [string] Record data format - Format to use to serialize events before
```

```
writing to Kafka.
    compression: # [string] Compression - Codec to use to compress the data before
sending to Kafka.
    maxRecordSizeKB: # [number] Max record size (KB, uncompressed) - Maximum size
(KB) of each record batch before compression. Setting should be < message.max.bytes
settings in Kafka brokers.
    flushEventCount: # [number] Max events per batch - Maximum number of events in a
batch before forcing a flush.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
Small values could cause the payload size to be smaller than the configured Max
record size.
    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
Confluent Schema Registry, i.e.: http://localhost:8081
      defaultKeySchemaId: # [number] Default key schema ID - Used when
__keySchemaIdOut is not present, to transform key values, leave blank if key
transformation is not required by default.
      defaultValueSchemaId: # [number] Default value schema ID - Used when
__valueSchemaIdOut is not present, to transform _raw, leave blank if value
transformation is not required by default.
      tls: # [object] TLS settings (client side)
        disabled: # [boolean] Disabled

        # -------------- if disabled is false ---------------

        rejectUnauthorized: # [boolean] Validate server certs - Reject certs that
are not authorized by a CA in the CA certificate path, or by another trusted CA
(e.g., the system's CA). Defaults to No.
        servername: # [string] Server name (SNI) - Server name for the SNI (Server
Name Indication) TLS extension. It must be a host name, and not an IP address.
        certificateName: # [string] Certificate name - The name of the predefined
certificate.
        caPath: # [string] CA certificate path - Path on client in which to find CA
certificates to verify the server's cert. PEM format. Can reference $ENV_VARS.
        privKeyPath: # [string] Private key path (mutual auth) - Path on client in
which to find the private key to use. PEM format. Can reference $ENV_VARS.
        certPath: # [string] Certificate path (mutual auth) - Path on client in
which to find certificates to use. PEM format. Can reference $ENV_VARS.
        passphrase: # [string] Passphrase - Passphrase to use to decrypt private
key.
        minVersion: # [string] Minimum TLS version - Minimum TLS version to use when
connecting
        maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when
connecting

        # ----------------------------------------------------------


      # ----------------------------------------------------------

    connectionTimeout: # [number] Connection timeout (ms) - Maximum time to wait for
a successful connection.
    requestTimeout: # [number] Request timeout (ms) - Maximum time to wait for a
successful request.
    sasl: # [object] Authentication - Authentication parameters to use when
connecting to brokers. Using TLS is highly recommended.
```

```
      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
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to No.
      servername: # [string] Server name (SNI) - Server name for the SNI (Server
Name Indication) TLS extension. It must be a host name, and not an IP address.
      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      caPath: # [string] CA certificate path - Path on client in which to find CA
certificates to verify the server's cert. PEM format. Can reference $ENV_VARS.
      privKeyPath: # [string] Private key path (mutual auth) - Path on client in
which to find the private key to use. PEM format. Can reference $ENV_VARS.
      certPath: # [string] Certificate path (mutual auth) - Path on client in which
to find certificates to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to use when
connecting
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when
connecting

      # ----------------------------------------------------

    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ----------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
```

```
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  confluent_cloud_output: # [object]
    type: # [string] Output Type
    brokers: # [array of strings] Brokers - List of Confluent Cloud brokers to
connect to, e.g., yourAccount.confluent.cloud:9092.
    tls: # [object] TLS settings (client side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to No.
      servername: # [string] Server name (SNI) - Server name for the SNI (Server
Name Indication) TLS extension. It must be a host name, and not an IP address.
      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      caPath: # [string] CA certificate path - Path on client in which to find CA
certificates to verify the server's cert. PEM format. Can reference $ENV_VARS.
      privKeyPath: # [string] Private key path (mutual auth) - Path on client in
which to find the private key to use. PEM format. Can reference $ENV_VARS.
      certPath: # [string] Certificate path (mutual auth) - Path on client in which
to find certificates to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to use when
connecting
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when
connecting

      # ----------------------------------------------------------

    topic: # [string] [required] Topic - The topic to publish events to. Can be
overridden using the __topicOut field.
    ack: # [number] Acknowledgments - Control the number of required
acknowledgments.
    format: # [string] Record data format - Format to use to serialize events before
writing to Kafka.
    compression: # [string] Compression - Codec to use to compress the data before
sending to Kafka.
    maxRecordSizeKB: # [number] Max record size (KB, uncompressed) - Maximum size
(KB) of each record batch before compression. Setting should be < message.max.bytes
settings in Kafka brokers.
    flushEventCount: # [number] Max events per batch - Maximum number of events in a
batch before forcing a flush.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
Small values could cause the payload size to be smaller than the configured Max
record size.
    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
Confluent Schema Registry, i.e.: http://localhost:8081
      defaultKeySchemaId: # [number] Default key schema ID - Used when
__keySchemaIdOut is not present, to transform key values, leave blank if key
```

```
transformation is not required by default.
      defaultValueSchemaId: # [number] Default value schema ID - Used when
__valueSchemaIdOut is not present, to transform _raw, leave blank if value
transformation is not required by default.
      tls: # [object] TLS settings (client side)
        disabled: # [boolean] Disabled

        # -------------- if disabled is false ---------------

        rejectUnauthorized: # [boolean] Validate server certs - Reject certs that
are not authorized by a CA in the CA certificate path, or by another trusted CA
(e.g., the system's CA). Defaults to No.
        servername: # [string] Server name (SNI) - Server name for the SNI (Server
Name Indication) TLS extension. It must be a host name, and not an IP address.
        certificateName: # [string] Certificate name - The name of the predefined
certificate.
        caPath: # [string] CA certificate path - Path on client in which to find CA
certificates to verify the server's cert. PEM format. Can reference $ENV_VARS.
        privKeyPath: # [string] Private key path (mutual auth) - Path on client in
which to find the private key to use. PEM format. Can reference $ENV_VARS.
        certPath: # [string] Certificate path (mutual auth) - Path on client in
which to find certificates to use. PEM format. Can reference $ENV_VARS.
        passphrase: # [string] Passphrase - Passphrase to use to decrypt private
key.
        minVersion: # [string] Minimum TLS version - Minimum TLS version to use when
connecting
        maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when
connecting

        # ----------------------------------------------------------


      # ------------------------------------------------------------

    connectionTimeout: # [number] Connection timeout (ms) - Maximum time to wait for
a successful connection.
    requestTimeout: # [number] Request timeout (ms) - Maximum time to wait for a
successful request.
    sasl: # [object] Authentication - Authentication parameters to use when
connecting to brokers. Using TLS is highly recommended.
      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
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
```

```
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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
cluster to send events to â e.g., http://elastic:9200/_bulk
    useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS
lookup. When a DNS server returns multiple addresses, this will cause Stream to
cycle through them in the order returned.

    # ------------------------------------------------------


    # -------------- if loadBalanced is true ---------------

    excludeSelf: # [boolean] Exclude current host IPs - Exclude all IPs of the
current host from the list of any resolved hostnames.
    urls: # [array] Bulk API URLs
      - url: # [string] URL - URL to an Elastic node to send events to â e.g.,
http://elastic:9200/_bulk
        weight: # [number] Load Weight - The weight to use for load-balancing
purposes.
    dnsResolvePeriodSec: # [number] DNS resolution period (seconds) - Re-resolve any
hostnames every this many seconds and pick up destinations from A records.
    loadBalanceStatsPeriodSec: # [number] Load balance stats period (seconds) - How
far back in time to keep traffic stats for load balancing purposes.

    # ------------------------------------------------------

    index: # [string] Index or Data Stream - Index or Data Stream to send events to.
Must be a JavaScript expression (which can evaluate to a constant value), enclosed
in quotes or backticks. Can be overwritten by an event's __index field.
    docType: # [string] Type - Document type to use for events. Can be overwritten
by an event's __type field
    concurrency: # [number] Request concurrency - Maximum number of ongoing requests
before blocking.
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the
request body.
    maxPayloadEvents: # [number] Max events per request - Max number of events to
```

```
include in the request body. Default is 0 (unlimited).
    compress: # [boolean] Compress - Whether to compress the payload body before
sending.
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to No.
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
a request to complete before aborting it.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
Small values could cause the payload size to be smaller than the configured Max body
size.
    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
which data should be logged when a request fails. Defaults to None.  All headers are
redacted by default, except those listed under `Safe Headers`.
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe
to log in plain text.
    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
select a stored secret

      # ---------------------------------------------------------

    elasticVersion: # [string] Elastic Version - Optional Elasticsearch version,
used to format events. If not specified, will auto-discover version.
    elasticPipeline: # [string] Elastic pipeline - Optional Elasticsearch
destination pipeline
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ---------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
```

```
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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.:
observability, access_log. The event's 'sourcetype' field (if set) will override
this value.
    messageField: # [string] Log message field - Name of field to send as log
message value. If not present, event will be serialized and sent as JSON.
    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,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    concurrency: # [number] Request concurrency - Maximum number of ongoing requests
before blocking.
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the
request body.
    maxPayloadEvents: # [number] Max events per request - Max number of events to
include in the request body. Default is 0 (unlimited).
    compress: # [boolean] Compress - Whether to compress the payload body before
sending.
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to Yes.
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
a request to complete before aborting it.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
Small values could cause the payload size to be smaller than the configured Max body
size.
    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
lookup. When a DNS server returns multiple addresses, this will cause Stream to
cycle through them in the order returned.
    failedRequestLoggingMode: # [string] Failed request logging mode - Determines
which data should be logged when a request fails. Defaults to None.  All headers are
redacted by default, except those listed under `Safe Headers`.
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe
to log in plain text.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
```

```
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ------------------------------------------------------

    authType: # [string] Authentication method - Enter API key directly, or select a
stored secret
    apiKey: # [string] API key - New Relic API key. Can be overridden using
__newRelic_apiKey field.

    # -------------- if authType is manual ---------------


    # ------------------------------------------------------

    textSecret: # [string] API key (text secret) - Select (or create) a stored text
secret

    # -------------- if authType is secret ---------------


    # ------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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
present in an event. For more information, see [here]
(https://docs.newrelic.com/docs/telemetry-data-platform/custom-data/custom-
events/data-requirements-limits-custom-event-data/#reserved-words).
    concurrency: # [number] Request concurrency - Maximum number of ongoing requests
before blocking.
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the
request body.
    maxPayloadEvents: # [number] Max events per request - Max number of events to
include in the request body. Default is 0 (unlimited).
    compress: # [boolean] Compress - Whether to compress the payload body before
sending.
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to Yes.
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
a request to complete before aborting it.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
Small values could cause the payload size to be smaller than the configured Max body
size.
    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
lookup. When a DNS server returns multiple addresses, this will cause Stream to
cycle through them in the order returned.
    failedRequestLoggingMode: # [string] Failed request logging mode - Determines
which data should be logged when a request fails. Defaults to None.  All headers are
redacted by default, except those listed under `Safe Headers`.
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe
to log in plain text.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ----------------------------------------------------------

    authType: # [string] Authentication method - Enter API key directly, or select a
stored secret
    apiKey: # [string] API key - New Relic API key. Can be overridden using
__newRelic_apiKey field.

    # -------------- if authType is manual ---------------


    # ----------------------------------------------------------

    textSecret: # [string] API key (text secret) - Select (or create) a stored text
secret

    # -------------- if authType is secret ---------------


    # ----------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  influxdb_output: # [object]
```

```
    type: # [string] Output Type
    url: # [string] Write API URL - URL of an InfluxDB cluster to send events to,
e.g., http://localhost:8086/write
    useV2API: # [boolean] Use v2 API - The v2 API can be enabled with InfluxDB
versions 1.8 and later.

    # -------------- 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
supplied Unix time values. Defaults to milliseconds.
    dynamicValueFieldName: # [boolean] Dynamic value fields - Enabling this will
pull the value field from the metric name. E,g, 'db.query.user' will use 'db.query'
as the measurement and 'user' as the value field.
    valueFieldName: # [string] Value field name - Name of the field in which to
store the metric when sending to InfluxDB. If dynamic generation is enabled and
fails, this will be used as a fallback.
    concurrency: # [number] Request concurrency - Maximum number of ongoing requests
before blocking.
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the
request body.
    maxPayloadEvents: # [number] Max events per request - Max number of events to
include in the request body. Default is 0 (unlimited).
    compress: # [boolean] Compress - Whether to compress the payload body before
sending.
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to No.
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
a request to complete before aborting it.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
Small values could cause the payload size to be smaller than the configured Max body
size.
    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
lookup. When a DNS server returns multiple addresses, this will cause Stream to
cycle through them in the order returned.
    failedRequestLoggingMode: # [string] Failed request logging mode - Determines
which data should be logged when a request fails. Defaults to None.  All headers are
redacted by default, except those listed under `Safe Headers`.
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe
to log in plain text.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------
```

```
    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    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 references your credentials

    # -------------------------------------------------------


    # -------------- if authType is textSecret ---------------

    textSecret: # [string] Token (text secret) - Select (or create) a stored text
secret

    # -------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  cloudwatch_output: # [object]
    type: # [string] Output Type
```

```
    logGroupName: # [string] Log group name - CloudWatch log group to associate
events with
    logStreamName: # [string] [required] Log stream prefix - Prefix for CloudWatch
log stream name. This prefix will be used to generate a unique log stream name per
cribl instance, for example: myStream_myHost_myOutputId
    awsAuthenticationMethod: # [string] Authentication method - AWS authentication
method. Choose Auto to use IAM roles.

    # -------------- 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
references your access key and secret key.

    # --------------------------------------------------------

    awsSecretKey: # [string] Secret key - Secret key
    region: # [string] [required] Region - Region where the CloudWatchLogs is
located
    endpoint: # [string] Endpoint - CloudWatchLogs service endpoint. If empty,
defaults to AWS' Region-specific endpoint. Otherwise, it must point to
CloudWatchLogs-compatible endpoint.
    signatureVersion: # [string] Signature version - Signature version to use for
signing CloudWatchLogs requests.
    reuseConnections: # [boolean] Reuse connections - Whether to reuse connections
between requests, which can improve performance.
    rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to
reject certificates that cannot be verified against a valid CA (e.g., self-signed
certificates).
    enableAssumeRole: # [boolean] Enable for CloudWatchLogs - Use Assume Role
credentials to access CloudWatchLogs
    assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the
role to assume
    assumeRoleExternalId: # [string] External ID - External ID to use when assuming
role
    maxQueueSize: # [number] Max queue size - Maximum number of queued batches
before blocking
    maxRecordSizeKB: # [number] Max record size (KB, uncompressed) - Maximum size
(KB) of each individual record before compression. For non compressible data 1MB is
the max recommended size
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
Small values could cause the payload size to be smaller than the configured Max
record size.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
```

```
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  minio_output: # [object]
    type: # [string] Output Type
    endpoint: # [string] [required] MinIO endpoint - MinIO service url (e.g.
http://minioHost:9000)
    bucket: # [string] MinIO bucket name - Name of the destination MinIO bucket.
This value can be a constant or a JavaScript expression that can only be evaluated
at init time. E.g. referencing a Global Variable: `myBucket-${C.vars.myVar}`.
    awsAuthenticationMethod: # [string] Authentication method - AWS authentication
method

    # -------------- 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
references your access key and secret key.

    # ------------------------------------------------------

    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
to buffer files, before compressing and moving to final destination. Use performant
stable storage.
    addIdToStagePath: # [boolean] Add output ID - Append output's ID to staging
location.
    removeEmptyDirs: # [boolean] Remove staging dirs - Remove empty staging
directories after moving files.

    # -------------- if removeEmptyDirs is true ---------------

    emptyDirCleanupSec: # [number] Staging cleanup period - How often (secs) to
clean-up empty directories when 'Remove Staging Dirs' is enabled.

    # ------------------------------------------------------
```

```
    destPath: # [string] [required] Key prefix - Root directory to prepend to path
before uploading. Enter a constant, or a JS expression enclosed in quotes or
backticks.
    signatureVersion: # [string] Signature version - Signature version to use for
signing MinIO requests.
    objectACL: # [string] Object ACL - Object ACL to assign to uploaded objects.
    storageClass: # [string] Storage class - Storage class to select for uploaded
objects.
    serverSideEncryption: # [string] Server-side encryption - Server-side encryption
for uploaded objects.
    reuseConnections: # [boolean] Reuse connections - Whether to reuse connections
between requests, which can improve performance.
    rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to
reject certificates that cannot be verified against a valid CA (e.g., self-signed
certificates).
    partitionExpr: # [string] Partitioning expression - JS expression defining how
files are partitioned and organized. Default is date-based. If blank, Stream will
fall back to the event's __partition field value â if present â otherwise to
each location's root directory.
    format: # [string] Data format - Format of the output data.

    # -------------- if format is json ---------------

    compress: # [string] Compress - Choose data compression format to apply before
moving files to final destination.

    # ------------------------------------------------------


    # -------------- if format is parquet ---------------

    parquetSchema: # [string] Parquet schema - Select a Parquet schema. New schemas
can be uploaded under Processing > Knowledge > Parquet Schemas
    parquetRowGroupSize: # [string] Row group size - Ideal memory size for row group
segments. E.g., 128MB or 1GB. Affects memory use when writing. Imposes a target, not
a strict limit; row groups final size may be larger or smaller.
    parquetPageSize: # [string] Page size - Ideal memory size for page segments.
E.g., 1MB or 128MB. Generally, lower values improve reading speed, while higher
values improve compression. Imposes a target, not a strict limit; pages final size
may be larger or smaller.
    spacer: # [null]
    parquetVersion: # [string] Parquet version - Determines which data types are
supported and how they are represented.
    parquetDataPageVersion: # [string] Data page version - Serialization format of
data pages. Note that not all reader implemtations support Data page V2.
    shouldLogInvalidRows: # [boolean] Log invalid rows - Output up to 20 unique rows
that were skipped due to data format mismatch. Must have Logging set to Debug to see
output.

    # ------------------------------------------------------

    baseFileName: # [string] File name prefix expression - JavaScript expression to
define the output filename prefix (can be constant).
    fileNameSuffix: # [string] File name suffix expression - JavaScript expression
to define the output filename suffix (can be constant).  The `__format` variable
refers to the value of the `Data format` field (`json` or `raw`).  The
`__compression` field refers to the kind of compression being used (`none` or
`gzip`)
    maxFileSizeMB: # [number] Max file size (MB) - Maximum uncompressed output file
```

```
size. Files of this size will be closed and moved to final output location.
    maxFileOpenTimeSec: # [number] 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.
    maxFileIdleTimeSec: # [number] 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.
    maxOpenFiles: # [number] Max open files - Maximum number of files to keep open
concurrently. When over, the oldest open files will be closed and moved to final
output location.
    onBackpressure: # [string] Backpressure behavior - Whether to block or drop
events when all receivers are exerting backpressure.
    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  statsd_output: # [object]
    type: # [string] Output Type
    protocol: # [string] Destination Protocol - Protocol to use when communicating
with the destination.

    # -------------- if protocol is tcp ---------------

    throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to
throttle while writing to an output. Also takes values with multiple-byte units,
such as KB, MB, GB, etc. (E.g., 42 MB.) Default value of 0 specifies no throttling.
    connectionTimeout: # [number] Connection Timeout - Amount of time (milliseconds)
to wait for the connection to establish before retrying
    writeTimeout: # [number] Write Timeout - Amount of time (milliseconds) to wait
for a write to complete before assuming connection is dead
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # ------------------------------------------------------

    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
the maximum size of packets sent to the destination. Also known as the MTU for the
network path to the destination system.
    flushPeriodSec: # [number] Flush period (sec) - Used when Protocol is TCP, to
specify how often buffers should be flushed resulting in records sent to the
destination.
    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  statsd_ext_output: # [object]
    type: # [string] Output Type
    protocol: # [string] Destination Protocol - Protocol to use when communicating
```

```
with the destination.

    # -------------- if protocol is tcp ---------------

    throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to
throttle while writing to an output. Also takes values with multiple-byte units,
such as KB, MB, GB, etc. (E.g., 42 MB.) Default value of 0 specifies no throttling.
    connectionTimeout: # [number] Connection Timeout - Amount of time (milliseconds)
to wait for the connection to establish before retrying
    writeTimeout: # [number] Write Timeout - Amount of time (milliseconds) to wait
for a write to complete before assuming connection is dead
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # ------------------------------------------------------

    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
the maximum size of packets sent to the destination. Also known as the MTU for the
network path to the destination system.
    flushPeriodSec: # [number] Flush period (sec) - Used when Protocol is TCP, to
specify how often buffers should be flushed resulting in records sent to the
destination.
    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  graphite_output: # [object]
    type: # [string] Output Type
    protocol: # [string] Destination Protocol - Protocol to use when communicating
with the destination.

    # -------------- if protocol is tcp ---------------

    throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to
throttle while writing to an output. Also takes values with multiple-byte units,
such as KB, MB, GB, etc. (E.g., 42 MB.) Default value of 0 specifies no throttling.
    connectionTimeout: # [number] Connection Timeout - Amount of time (milliseconds)
to wait for the connection to establish before retrying
    writeTimeout: # [number] Write Timeout - Amount of time (milliseconds) to wait
for a write to complete before assuming connection is dead
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # ------------------------------------------------------

    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
the maximum size of packets sent to the destination. Also known as the MTU for the
network path to the destination system.
    flushPeriodSec: # [number] Flush period (sec) - Used when Protocol is TCP, to
specify how often buffers should be flushed resulting in records sent to the
destination.
```

```
    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  router_output: # [object]
    type: # [string] Output Type
    rules: # [array] Rules - Event routing rules
      - filter: # [string] Filter Expression - JavaScript expression to select
events to send to output
        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
being checked against other rules
    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  sqs_output: # [object]
    type: # [string] Output Type
    queueName: # [string] Queue Name - The name, URL, or ARN of the SQS queue to
send events to. When a non-AWS URL is specified, format must be:
'{url}/myQueueName'. E.g., '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 init time. E.g., referencing a Global Variable:
`https://host:port/myQueue-${C.vars.myVar}`.
    queueType: # [string] [required] Queue Type - The queue type used (or created).
Defaults to Standard.
    awsAccountId: # [string] AWS Account ID - SQS queue owner's AWS account ID.
Leave empty if SQS queue is in same AWS account.
    messageGroupId: # [string] Message Group ID - This parameter applies only to
FIFO queues. The tag that specifies that a message belongs to a specific message
group. Messages that belong to the same message group are processed in a FIFO
manner. Use event field __messageGroupId to override this value.
    createQueue: # [boolean] Create Queue - Create queue if it does not exist.
    awsAuthenticationMethod: # [string] Authentication method - AWS authentication
method. Choose Auto to use IAM roles.

    # -------------- 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
references your access key and secret key.

    # --------------------------------------------------------
```

```
    awsSecretKey: # [string] Secret key - Secret key
    region: # [string] Region - AWS Region where the SQS queue is located. Required,
unless the Queue entry is a URL or ARN that includes a Region.
    endpoint: # [string] Endpoint - SQS service endpoint. If empty, defaults to AWS'
Region-specific endpoint. Otherwise, it must point to SQS-compatible endpoint.
    signatureVersion: # [string] Signature version - Signature version to use for
signing SQS requests.
    reuseConnections: # [boolean] Reuse connections - Whether to reuse connections
between requests, which can improve performance.
    rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to
reject certificates that cannot be verified against a valid CA (e.g., self-signed
certificates).
    enableAssumeRole: # [boolean] Enable for SQS - Use Assume Role credentials to
access SQS
    assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the
role to assume
    assumeRoleExternalId: # [string] External ID - External ID to use when assuming
role
    maxQueueSize: # [number] Max queue size - Maximum number of queued batches
before blocking.
    maxRecordSizeKB: # [number] Max record size (KB) - Maximum size (KB) of batches
to send. Per the SQS spec, the max allowed value is 256 KB.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
Small values could cause the payload size to be smaller than the configured Max
record size.
    maxInProgress: # [number] Max concurrent requests - The maximum number of in-
progress API requests before backpressure is applied.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # --------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  snmp_output: # [object]
```

```
    type: # [string] Output Type
    hosts: # [array] SNMP Trap Destinations - One or more SNMP destinations to
forward traps to
      - host: # [string] Address - Destination host
        port: # [number] Port - Destination port, default is 162
    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  sumo_logic_output: # [object]
    type: # [string] Output Type
    url: # [string] API URL - Sumo Logic HTTP collector URL to which events should
be sent.
    customSource: # [string] Custom source name - Optionally, override the source
name configured on the SumoÂ Logic HTTP collector. This can also be overridden at
the event level with the __sourceName field.
    customCategory: # [string] Custom source category - Optionally, override the
source category configured on the SumoÂ Logic HTTP collector. This can also be
overridden at the event level with the __sourceCategory field.
    concurrency: # [number] Request concurrency - Maximum number of ongoing requests
before blocking.
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the
request body.
    maxPayloadEvents: # [number] Max events per request - Max number of events to
include in the request body. Default is 0 (unlimited).
    compress: # [boolean] Compress - Whether to compress the payload body before
sending.
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to Yes.
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
a request to complete before aborting it.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
Small values could cause the payload size to be smaller than the configured Max body
size.
    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
lookup. When a DNS server returns multiple addresses, this will cause Stream to
cycle through them in the order returned.
    failedRequestLoggingMode: # [string] Failed request logging mode - Determines
which data should be logged when a request fails. Defaults to None.  All headers are
redacted by default, except those listed under `Safe Headers`.
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe
to log in plain text.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
```

```
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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
message to send. If not specified, Stream sends a JSON representation of the whole
event.
    source: # [string] Source - Name of the source to send with logs. When you send
logs as JSON objects, the event's 'source' field (if set) will override this value.
    host: # [string] Host - Name of the host to send with logs. When you send logs
as JSON objects, the event's 'host' field (if set) will override this value.
    service: # [string] Service - Name of the service to send with logs. When you
send logs as JSON objects, the event's '__service' field (if set) will override this
value.
    tags: # [array of strings] Datadog tags - List of tags to send with logs (e.g.,
'env:prod', 'env_staging:east').
    allowApiKeyFromEvents: # [boolean] Allow API key from events - If enabled, the
API key can be set from the event's '__agent_api_key' field.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
    severity: # [string] Severity - Default value for message severity. When you
send logs as JSON objects, the event's '__severity' field (if set) will override
this value.
    site: # [string] Datadog site
    concurrency: # [number] Request concurrency - Maximum number of ongoing requests
before blocking.
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the
request body.
    maxPayloadEvents: # [number] Max events per request - Max number of events to
include in the request body. Default is 0 (unlimited).
    compress: # [boolean] Compress - Whether to compress the payload body before
sending.
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to Yes.
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
a request to complete before aborting it.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
```

```
Small values could cause the payload size to be smaller than the configured Max body
size.
    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
lookup. When a DNS server returns multiple addresses, this will cause Stream to
cycle through them in the order returned.
    failedRequestLoggingMode: # [string] Failed request logging mode - Determines
which data should be logged when a request fails. Defaults to None.  All headers are
redacted by default, except those listed under `Safe Headers`.
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe
to log in plain text.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # --------------------------------------------------------

    authType: # [string] Authentication method - Enter API key directly, or select a
stored secret
    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
secret

    # -------------- if authType is secret ---------------


    # --------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
  grafana_cloud_output: # [object]
```

```
    type: # [string] Output Type
    lokiUrl: # [string] Loki URL - The endpoint to send logs to, e.g.: https://logs-prod-us-central1.grafana.net
    prometheusUrl: # [string] [required] Prometheus URL - The remote_write endpoint to send Prometheus metrics to, e.g.: https://prometheus-blocks-prod-us-central1.grafana.net/api/prom/push
    message: # [string] Logs message field - Name of the event field that contains the message to send. If not specified, Stream sends a JSON representation of the whole event.
    messageFormat: # [string] Message Format - Which format to use when sending logs to Loki (Protobuf or JSON).  Defaults to Protobuf.

    # -------------- if messageFormat is json ---------------

    compress: # [boolean] Compress - Whether to compress the payload body before sending. Applies only to Loki's JSON payloads, as both Prometheus' and Loki's Protobuf variant are snappy-compressed by default.

    # --------------------------------------------------------

    labels: # [array] Logs labels - List of labels to send with logs. Labels define Loki streams, so use static labels to avoid proliferating label value combinations and streams. Can be merged and/or overridden by the event's __labels field (e.g.: '__labels: {host: "cribl.io", level: "error"}').
      - name: # [string] Name - Name of the label.
        value: # [string] Value - Value of the label.
    metricRenameExpr: # [string] Metrics renaming expression - A JS expression that can be used to rename metrics. E.g.: name.replace(/\./g, '_') will replace all '.' characters in a metric's name with the supported '_' character. Use the 'name' global variable to access the metric's name.  You can access event fields' values via __e.<fieldName>.
    prometheusAuth: # [object]
      authType: # [string] Authentication Type - The authentication method to use for the HTTP requests

      # -------------- if authType is token ---------------

      token: # [string] Auth token - Bearer token to include in the authorization header. In Grafana Cloud, this is generally built by concatenating the username and the API key, separated by a colon. E.g.: <your-username>:<your-api-key>.

      # --------------------------------------------------------


      # -------------- if authType is textSecret ---------------

      textSecret: # [string] Auth token (text secret) - Select (or create) a stored text secret

      # --------------------------------------------------------


      # -------------- if authType is basic ---------------

      username: # [string] Username - Username for authentication
      password: # [string] Password - Password (a.k.a API key in Grafana Cloud domain) for authentication

      # --------------------------------------------------------
```

```
      # -------------- if authType is credentialsSecret ---------------

      credentialsSecret: # [string] Credentials secret - Select (or create) a secret
that references your credentials

      # ----------------------------------------------------------

    lokiAuth: # [object]
      authType: # [string] Authentication Type - The authentication method to use
for the HTTP requests

      # -------------- if authType is token ---------------

      token: # [string] Auth token - Bearer token to include in the authorization
header. In Grafana Cloud, this is generally built by concatenating the username and
the API key, separated by a colon. E.g.: <your-username>:<your-api-key>.

      # ----------------------------------------------------------


      # -------------- if authType is textSecret ---------------

      textSecret: # [string] Auth token (text secret) - Select (or create) a stored
text secret

      # ----------------------------------------------------------


      # -------------- if authType is basic ---------------

      username: # [string] Username - Username for authentication
      password: # [string] Password - Password (a.k.a API key in Grafana Cloud
domain) for authentication

      # ----------------------------------------------------------


      # -------------- if authType is credentialsSecret ---------------

      credentialsSecret: # [string] Credentials secret - Select (or create) a secret
that references your credentials

      # ----------------------------------------------------------

    concurrency: # [number] Request concurrency - Maximum number of ongoing requests
before blocking. Warning: Setting this value > 1 can cause Loki and Prometheus to
complain about entries being delivered out of order.
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the
request body. Warning: Setting this too low can increase the number of ongoing
requests (depending on the value of 'Request concurrency'); this can cause Loki and
Prometheus to complain about entries being delivered out of order.
    maxPayloadEvents: # [number] Max events per request - Maximum number of events
to include in the request body. Default is 0 (unlimited). Warning: Setting this too
low can increase the number of ongoing requests (depending on the value of 'Request
concurrency'); this can cause Loki and Prometheus to complain about entries being
delivered out of order.
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to Yes.
```

```
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
a request to complete before aborting it.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
Small values could cause the payload size to be smaller than the configured Maximum
time between requests. Small values can reduce the payload size below the configured
'Max record size' and 'Max events per request'. Warning: Setting this too low can
increase the number of ongoing requests (depending on the value of 'Request
concurrency'); this can cause Loki and Prometheus to complain about entries being
delivered out of order.
    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
lookup. When a DNS server returns multiple addresses, this will cause Stream to
cycle through them in the order returned.
    failedRequestLoggingMode: # [string] Failed request logging mode - Determines
which data should be logged when a request fails. Defaults to None.  All headers are
redacted by default, except those listed under `Safe Headers`.
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe
to log in plain text.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ----------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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
the message to send. If not specified, Stream sends a JSON representation of the
whole event.
    messageFormat: # [string] Message Format - Which format to use when sending logs
to Loki (Protobuf or JSON).  Defaults to Protobuf.
```

```
    # -------------- if messageFormat is json ---------------

    compress: # [boolean] Compress - Whether to compress the payload body before
sending.

    # --------------------------------------------------------

    labels: # [array] Logs labels - List of labels to send with logs. Labels define
Loki streams, so use static labels to avoid proliferating label value combinations
and streams. Can be merged and/or overridden by the event's __labels field (e.g.:
'__labels: {host: "cribl.io", level: "error"}').
      - name: # [string] Name - Name of the label.
        value: # [string] Value - Value of the label.
    authType: # [string] Authentication Type - The authentication method to use for
the HTTP requests

    # -------------- if authType is token ---------------

    token: # [string] Auth token - Bearer token to include in the authorization
header. In Grafana Cloud, this is generally built by concatenating the username and
the API key, separated by a colon. E.g.: <your-username>:<your-api-key>.

    # --------------------------------------------------------


    # -------------- if authType is textSecret ---------------

    textSecret: # [string] Auth token (text secret) - Select (or create) a stored
text secret

    # --------------------------------------------------------


    # -------------- if authType is basic ---------------

    username: # [string] Username - Username for authentication
    password: # [string] Password - Password (a.k.a API key in Grafana Cloud domain)
for authentication

    # --------------------------------------------------------


    # -------------- if authType is credentialsSecret ---------------

    credentialsSecret: # [string] Credentials secret - Select (or create) a secret
that references your credentials

    # --------------------------------------------------------

    concurrency: # [number] Request concurrency - Maximum number of ongoing requests
before blocking. Warning: Setting this value > 1 can cause Loki to complain about
entries being delivered out of order.
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the
request body. Warning: Setting this too low can increase the number of ongoing
requests (depending on the value of 'Request concurrency'); this can cause Loki to
complain about entries being delivered out of order.
    maxPayloadEvents: # [number] Max events per request - Maximum number of events
to include in the request body. Defaults to 0 (unlimited). Warning: Setting this too
low can increase the number of ongoing requests (depending on the value of 'Request
concurrency'); this can cause Loki to complain about entries being delivered out of
```

```
order.
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to No.
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
a request to complete before aborting it.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
Small values could cause the payload size to be smaller than the configured Maximum
time between requests. Small values can reduce the payload size below the configured
'Max record size' and 'Max events per request'. Warning: Setting this too low can
increase the number of ongoing requests (depending on the value of 'Request
concurrency'); this can cause Loki to complain about entries being delivered out of
order.
    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
lookup. When a DNS server returns multiple addresses, this will cause Stream to
cycle through them in the order returned.
    failedRequestLoggingMode: # [string] Failed request logging mode - Determines
which data should be logged when a request fails. Defaults to None.  All headers are
redacted by default, except those listed under `Safe Headers`.
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe
to log in plain text.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ---------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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
can be used to rename metrics. E.g.: name.replace(/\./g, '_') will replace all '.'
```

```
characters in a metric's name with the supported '_' character. Use the 'name'
global variable to access the metric's name.  You can access event fields' values
via __e.<fieldName>.
    sendMetadata: # [boolean] Send metadata - Whether to generate and send metadata
(`type` and `metricFamilyName`) requests.

    # -------------- if sendMetadata is true ---------------

    metricsFlushPeriodSec: # [number] Metadata flush period (sec) - How frequently
metrics metadata is sent out. Value cannot be smaller than the base Flush period
(sec) set above.

    # -------------------------------------------------------

    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    concurrency: # [number] Request concurrency - Maximum number of ongoing requests
before blocking.
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the
request body.
    maxPayloadEvents: # [number] Max events per request - Max number of events to
include in the request body. Default is 0 (unlimited).
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to No.
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
a request to complete before aborting it.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
Small values could cause the payload size to be smaller than the configured Max body
size.
    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
lookup. When a DNS server returns multiple addresses, this will cause Stream to
cycle through them in the order returned.
    failedRequestLoggingMode: # [string] Failed request logging mode - Determines
which data should be logged when a request fails. Defaults to None.  All headers are
redacted by default, except those listed under `Safe Headers`.
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe
to log in plain text.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
```

```
    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
that references your credentials

    # -------------------------------------------------------


    # -------------- if authType is textSecret ---------------

    textSecret: # [string] Token (text secret) - Select (or create) a stored text
secret

    # -------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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
files are partitioned and organized. If left blank, Cribl Stream will fallback on
event.__partition.
    maxDataSize: # [string] Max data size - Maximum disk space allowed to be
consumed (e.g., 420MB or 4GB). Once reached, older data will be deleted.
    maxDataTime: # [string] Max data age - Maximum amount of time to retain data
(e.g., 2h or 4d). Once reached, older data will be deleted.
    compress: # [string] Compression - Select data compression format. Optional.
    destPath: # [string] Path location - Path to use to write metrics. Defaults to
$CRIBL_HOME/state/<id>`
    onBackpressure: # [string] Backpressure behavior - Whether to block or drop
events when all receivers are exerting backpressure.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
```

```
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
  open_telemetry_output: # [object]
    type: # [string] Output Type
    endpoint: # [string] Endpoint - The endpoint to send OTEL events to. It can be
any valid URL or an IP address (both IPv4 and IPv6 are supported). If the port is
not specified, it will default to 4317, unless the endpoint is an HTTPS-based URL or
TLS is enabled. In such cases, it will default to 443.
    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
that references your credentials

    # ---------------------------------------------------------


    # -------------- if authType is textSecret ---------------

    textSecret: # [string] Token (text secret) - Select (or create) a stored text
secret

    # ---------------------------------------------------------

    tls: # [object] TLS settings (client side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to No.
      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      caPath: # [string] CA certificate path - Path on client in which to find CA
certificates to verify the server's cert. PEM format. Can reference $ENV_VARS.
      privKeyPath: # [string] Private key path (mutual auth) - Path on client in
which to find the private key to use. PEM format. Can reference $ENV_VARS.
      certPath: # [string] Certificate path (mutual auth) - Path on client in which
to find certificates to use. PEM format. Can reference $ENV_VARS.
```

```
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to use when
connecting
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when
connecting

      # ------------------------------------------------------

    metadata: # [array] Metadata - Extra information to send with each gRPC request
in the form of a list of key-value pairs. Value supports JavaScript expressions that
are evaluated just once, when the destination gets started. In case you need to pass
credentials as metadata, it's encouraged to use 'C.Secret'.
      - 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
before blocking.
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the
request body.
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
a request to complete before aborting it.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
Small values could cause the payload size to be smaller than the configured Max body
size.
    failedRequestLoggingMode: # [string] Failed request logging mode - Determines
which data should be logged when a request fails. Defaults to None.  All headers are
redacted by default, except those listed under `Safe Headers`.
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe
to log in plain text.
    connectionTimeout: # [number] Connection Timeout - Amount of time (milliseconds)
to wait for the connection to establish before retrying
    keepAliveTime: # [number] Keep Alive Time (seconds) - How often the sender
should ping the peer to keep the connection alive.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
```

```
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  dataset_output: # [object]
    type: # [string] Output Type
    messageField: # [string] Message field - Name of the event field that contains
the message or attributes to send. If not specified, all of the event's non-internal
fields will be sent as attributes.
    excludeFields: # [array of strings] Exclude fields - Fields to exclude from the
event if the Message field is either unspecified or refers to an object. Ignored if
the Message field is a string. If empty, we send all non-internal fields.
    serverHostField: # [string] Server/host field - Name of the event field that
contains the `serverHost` identifier. If not specified, defaults to
`cribl_<outputId>`.
    timestampField: # [string] Timestamp field - Name of the event field that
contains the timestamp. If not specified, defaults to `ts`, `_time`, or
`Date.now()`, in that order.
    defaultSeverity: # [string] Severity - Default value for event severity. If the
`sev` or `__severity` fields are set on an event, the first one matching will
override this value.
    site: # [string] DataSet site - DataSet site to which events should be sent
    customUrl: # [string]
    concurrency: # [number] Request concurrency - Maximum number of ongoing requests
before blocking.
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the
request body.
    maxPayloadEvents: # [number] Max events per request - Max number of events to
include in the request body. Default is 0 (unlimited).
    compress: # [boolean] Compress - Whether to compress the payload body before
sending.
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to Yes.
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
a request to complete before aborting it.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
Small values could cause the payload size to be smaller than the configured Max body
size.
    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
lookup. When a DNS server returns multiple addresses, this will cause Stream to
cycle through them in the order returned.
    failedRequestLoggingMode: # [string] Failed request logging mode - Determines
which data should be logged when a request fails. Defaults to None.  All headers are
redacted by default, except those listed under `Safe Headers`.
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe
to log in plain text.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
```

```
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ------------------------------------------------------

    authType: # [string] Authentication method - Enter API key directly, or select a
stored secret
    apiKey: # [string] API key - A 'Log Write Access' API key for the DataSet
account

    # -------------- if authType is manual ---------------


    # ------------------------------------------------------

    textSecret: # [string] API key (text secret) - Select (or create) a stored text
secret

    # -------------- if authType is secret ---------------


    # ------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
  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
current host from the list of any resolved hostnames.
    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
disable TLS.
        servername: # [string] TLS Servername - Servername to use if establishing a
```

```
TLS connection. If not specified, defaults to connection host (iff not an IP);
otherwise, to the global TLS settings.
        weight: # [number] Load Weight - The weight to use for load-balancing
purposes.
    dnsResolvePeriodSec: # [number] DNS resolution period (seconds) - Re-resolve any
hostnames every this many seconds and pick up destinations from A records.
    loadBalanceStatsPeriodSec: # [number] Load balance stats period (seconds) - How
far back in time to keep traffic stats for load balancing purposes.
    maxConcurrentSenders: # [number] Max connections - Maximum number of concurrent
connections (per worker process). A random set of IPs will be picked on every DNS
resolution period. Use 0 for unlimited.

    # ------------------------------------------------------

    compression: # [string] Compression - Codec to use to compress the data before
sending
    throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to
throttle while writing to an output. Also takes values with multiple-byte units,
such as KB, MB, GB, etc. (E.g., 42 MB.) Default value of 0 specifies no throttling.
    tls: # [object] TLS settings (client side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to No.
      servername: # [string] Server name (SNI) - Server name for the SNI (Server
Name Indication) TLS extension. It must be a host name, and not an IP address.
      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      caPath: # [string] CA certificate path - Path on client in which to find CA
certificates to verify the server's cert. PEM format. Can reference $ENV_VARS.
      privKeyPath: # [string] Private key path (mutual auth) - Path on client in
which to find the private key to use. PEM format. Can reference $ENV_VARS.
      certPath: # [string] Certificate path (mutual auth) - Path on client in which
to find certificates to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to use when
connecting
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when
connecting

      # ------------------------------------------------------

    connectionTimeout: # [number] Connection Timeout - Amount of time (milliseconds)
to wait for the connection to establish before retrying
    writeTimeout: # [number] Write Timeout - Amount of time (milliseconds) to wait
for a write to complete before assuming connection is dead
    tokenTTLMinutes: # [number] Auth Token TTL minutes - The number of minutes
before the internally generated authentication token expires, valid values between 1
and 60
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
```

```
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ------------------------------------------------------

    authType: # [string] Authentication method - Enter a token directly, or provide
a secret referencing a token
    authToken: # [string] Auth token - Optional authentication token to include as
part of the connection header

    # -------------- if authType is manual ---------------


    # ------------------------------------------------------

    textSecret: # [string] Auth token (text secret) - Select (or create) a stored
text secret

    # -------------- if authType is secret ---------------


    # ------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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
current host from the list of any resolved hostnames.
    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
disable TLS.
        servername: # [string] TLS Servername - Servername to use if establishing a
TLS connection. If not specified, defaults to connection host (iff not an IP);
otherwise, to the global TLS settings.
        weight: # [number] Load Weight - The weight to use for load-balancing
purposes.
    dnsResolvePeriodSec: # [number] DNS resolution period (seconds) - Re-resolve any
hostnames every this many seconds and pick up destinations from A records.
    loadBalanceStatsPeriodSec: # [number] Load balance stats period (seconds) - How
far back in time to keep traffic stats for load balancing purposes.
    maxConcurrentSenders: # [number] Max connections - Maximum number of concurrent
connections (per worker process). A random set of IPs will be picked on every DNS
resolution period. Use 0 for unlimited.

    # ------------------------------------------------------

    compression: # [string] Compression - Codec to use to compress the data before
sending
    throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to
throttle while writing to an output. Also takes values with multiple-byte units,
such as KB, MB, GB, etc. (E.g., 42 MB.) Default value of 0 specifies no throttling.
    tls: # [object] TLS settings (client side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to No.
      servername: # [string] Server name (SNI) - Server name for the SNI (Server
Name Indication) TLS extension. It must be a host name, and not an IP address.
      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      caPath: # [string] CA certificate path - Path on client in which to find CA
certificates to verify the server's cert. PEM format. Can reference $ENV_VARS.
      privKeyPath: # [string] Private key path (mutual auth) - Path on client in
which to find the private key to use. PEM format. Can reference $ENV_VARS.
      certPath: # [string] Certificate path (mutual auth) - Path on client in which
to find certificates to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to use when
connecting
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when
connecting

      # ------------------------------------------------------

    connectionTimeout: # [number] Connection Timeout - Amount of time (milliseconds)
to wait for the connection to establish before retrying
    writeTimeout: # [number] Write Timeout - Amount of time (milliseconds) to wait
for a write to complete before assuming connection is dead
    tokenTTLMinutes: # [number] Auth Token TTL minutes - The number of minutes
before the internally generated authentication token expires, valid values between 1
and 60
    excludeFields: # [array of strings] Exclude Fields - Fields to exclude from the
event. By default, all internal fields except `__output` are sent.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.
```

```
    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ----------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  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.,
http://localhost:10200
    useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS
lookup. When a DNS server returns multiple addresses, this will cause Stream to
cycle through them in the order returned.

    # ----------------------------------------------------------


    # -------------- if loadBalanced is true ---------------

    excludeSelf: # [boolean] Exclude current host IPs - Exclude all IPs of the
current host from the list of any resolved hostnames.
    urls: # [array] Cribl Worker Endpoints
      - url: # [string] Cribl Endpoint - URL of a Cribl Worker to send events to,
e.g., http://localhost:10200
        weight: # [number] Load Weight - The weight to use for load-balancing
purposes.
    dnsResolvePeriodSec: # [number] DNS resolution period (seconds) - Re-resolve any
hostnames every this many seconds and pick up destinations from A records.
    loadBalanceStatsPeriodSec: # [number] Load balance stats period (seconds) - How
far back in time to keep traffic stats for load balancing purposes.

    # ----------------------------------------------------------

    tls: # [object] TLS settings (client side)
```

```
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to No.
      servername: # [string] Server name (SNI) - Server name for the SNI (Server
Name Indication) TLS extension. It must be a host name, and not an IP address.
      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      caPath: # [string] CA certificate path - Path on client in which to find CA
certificates to verify the server's cert. PEM format. Can reference $ENV_VARS.
      privKeyPath: # [string] Private key path (mutual auth) - Path on client in
which to find the private key to use. PEM format. Can reference $ENV_VARS.
      certPath: # [string] Certificate path (mutual auth) - Path on client in which
to find certificates to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to use when
connecting
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when
connecting

      # ----------------------------------------------------------

    tokenTTLMinutes: # [number] Auth Token TTL minutes - The number of minutes
before the internally generated authentication token expires, valid values between 1
and 60.
    excludeFields: # [array of strings] Exclude fields - Fields to exclude from the
event. By default, all internal fields except `__output` are sent.
    compression: # [string] Compression - Codec to use to compress the data before
sending.
    concurrency: # [number] Request concurrency - Maximum number of ongoing requests
before blocking.
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the
request body.
    maxPayloadEvents: # [number] Max events per request - Max number of events to
include in the request body. Default is 0 (unlimited).
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to Yes.
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
a request to complete before aborting it.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
Small values could cause the payload size to be smaller than the configured Max body
size.
    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
which data should be logged when a request fails. Defaults to None.  All headers are
redacted by default, except those listed under `Safe Headers`.
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe
to log in plain text.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------
```

```
    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
  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
to, e.g., https://cloud.us.humio.com/api/v1/ingest/hec for JSON and
https://cloud.us.humio.com/api/v1/ingest/hec/raw for raw
    useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS
lookup. When a DNS server returns multiple addresses, this will cause Stream to
cycle through them in the order returned.

    # ------------------------------------------------------


    # -------------- if loadBalanced is true ---------------

    excludeSelf: # [boolean] Exclude current host IPs - Exclude all IPs of the
current host from the list of any resolved hostnames.
    urls: # [array] Humio HEC Endpoints
      - url: # [string] HEC Endpoint - URL to a Humio HEC endpoint to send events
to, e.g., https://cloud.us.humio.com/api/v1/ingest/hec for JSON and
https://cloud.us.humio.com/api/v1/ingest/hec/raw for raw
        weight: # [number] Load Weight - The weight to use for load-balancing
purposes.
    dnsResolvePeriodSec: # [number] DNS resolution period (seconds) - Re-resolve any
hostnames every this many seconds and pick up destinations from A records.
    loadBalanceStatsPeriodSec: # [number] Load balance stats period (seconds) - How
far back in time to keep traffic stats for load balancing purposes.

    # ------------------------------------------------------

    concurrency: # [number] Request concurrency - Maximum number of ongoing requests
before blocking.
```

```
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the
request body.
    maxPayloadEvents: # [number] Max events per request - Max number of events to
include in the request body. Default is 0 (unlimited).
    compress: # [boolean] Compress - Whether to compress the payload body before
sending.
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to Yes.
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
a request to complete before aborting it.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
Small values could cause the payload size to be smaller than the configured Max body
size.
    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
which data should be logged when a request fails. Defaults to None.  All headers are
redacted by default, except those listed under `Safe Headers`.
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe
to log in plain text.
    format: # [string] Request Format - Send data in JSON format to the
api/v1/ingest/hec endpoint , or raw 1-request-per-line to the api/v1/ingest/hec/raw
endpoint .
    authType: # [string] Authentication method - Enter a token directly, or provide
a secret referencing a token

    # -------------- 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 text secret

    # -----------------------------------------------------

    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
```

```
    pqControls: # [object]

    # ------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
```

14.3.15. 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
order. If not specified parser will auto-generate.
```

14.3.16. 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
```

14.3.17. 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

```
egex_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.
```

14.3.18. 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
everything in the system.'
  policy:
    - '* *'
reader_all:
  description: 'Members with reader_all role get read-only access to all
Worker Groups/Fleets.'
  policy:
    - GroupRead *
collect_all:
  description: 'Members of this group can run existing collection jobs of all
Worker Groups/Fleets'
  policy:
    - GroupCollect *
editor_all:
  description: 'Members with editor_all role get read/write access to all
Worker Groups/Fleets.'
  policy:
    - GroupEdit *
owner_all:
  description: 'Members with owner_all role get read/write access as well as Deploy
permissions to all Worker Groups/Fleets.'
  policy:
    - GroupFull *
user:
  description: 'The base user role allows users to see the system info along with
their own profile settings.'
  policy:
    - GET /system/info
    - GET /system/info/*
    - GET /system/users
    - GET /system/instance/distributed
    - GET /system/instance/distributed/*
    - GET /clui
    - PATCH /ui/*
```

14.3.19. 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
with sample with. Select GLOBAL if not sure. Deprecated.
  description: # [string] Description - Brief description of this sample file.
Optional.
  ttl: # [number] Expiration (hours) - Time to live for the sample, the TTL is reset
after each use of the sample. Leave empty to never expire.
  tags: # [string] Tags - One or more tags related to this sample file. Optional.
```

The corresponding sample files reside in `$CRIBL_HOME/data/samples`.

14.3.20. 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-
09.
```

14.3.21. scripts.yml

`scripts.yml` stores configuration data for scripts configured at **Settings** > **Global Settings** > **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
script
  env: # [object] Env Variables - Extra environment variables to set when executing
script
```

14.3.22. 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.
```

14.4. Cribl Expressions

Native Cribl Edge function 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.

C.Crypto – Data Encryption and Decryption Functions

`C.Crypto.decrypt`
(method) `Crypto.decrypt(value: string, escape: boolean, escapeSeq: string): string`
Decrypt all occurrences of ciphers in the given value. Instances that cannot be decrypted (for any reason) are left intact.
@param – `value` – String in which to look for ciphers.
@param – `escape` – Boolean, defaults to `false`. Set to `true` to escape double quotes in output after decryption. (E.g., for data encrypted in Splunk.)
@param – `escapeSeq` – String used to escape double quotes. The default `'"'` escapes CSV output.
@returns – `value` with ciphers decrypted.

`C.Crypto.encrypt`
(method) `Crypto.encrypt(value: any, keyclass: number, keyId?: string, defaultVal?: string): string`
Encrypt the given value with the `keyId`, or with a `keyId` picked up automatically based on `keyclass`.
@param {string | Buffer} `value` – what to encrypt.
@param – `keyclass` – if `keyId` isn't specified, pick one at the given `keyclass`.
@param – `keyId` - encryption keyId, takes precedence over `keyclass`.
@param – `defaultVal` – what to return if encryption fails for any reason; if unspecified, the original value is returned.
@returns – if encryption succeeds, the encrypted value; otherwise, `defaultVal` if specified; otherwise, `value`.

`C.Crypto.createHmac`
(method) `Crypto.createHmac(value: string | Buffer, secret: string, algorithm: string = 'sha256', outputFormat: 'base64' | 'hex' | 'latin1' = 'hex'): string`
Generates an HMAC that can be added to events, or can be used to validate events that contain an HMAC. (Available in Cribl Edge v.3.1.2+.)
@param – `value` – The data to encrypt, as a string. (When the `outputFormat` is invalid or undefined, this parameter is returned as the digest, via a Buffer.)
@param} – `secret` – The secret key used to generate the MAC, as a string.
@param – `algorithm` – The hash algorithm used to generate the MAC, as a string. Defaults to `'sha256'`. Run `openssl list -digest-algorithms` to see the list of available algorithms.
@param – `outputFormat` – One of `'base64'`, `'hex'`, or `'latin1'`. Defaults to `'hex'`.
@returns – The calculated HMAC digest on success; otherwise, `value`.

C.Decode – Data Decoding Functions

`C.Decode.base64`
(method) `Decode.base64(val: string, resultEnc?: string): any`
Performs base64 decoding of the given string. Returns a string or Buffer, depending on the `resultEnc` value, which defaults to `'utf8'`.
@param – `val` – value to base64-decode.
@param – `resultEnc` – encoding to use to convert the binary data to a string. Defaults to `'utf8'`.
Use `'utf8-valid'` to validate that result is valid UTF8; use `'buffer'` if you need the binary data in a Buffer.

`C.Decode.gzip`
(method) `Decode.gzip(value: any, encoding?: string): string`
Gunzip the supplied value.
@param – `value` – the value to gunzip.
@param – `encoding` – 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.

`C.Decode.hex`
(method) `Decode.hex(val: string): number`
Performs hex to number conversion. (Returns `NaN` if value cannot be converted to a number.)
@param – `val` – hex string to parse to a number (e.g., "0xcafe").

`C.Decode.uri`
(method) `Decode.uri(val: string): string`
Performs URI-decoding of the given string.
@param – `val` – value to URI-decode.

C.Encode – Data Encoding Functions

`C.Encode.base64`
(method) `Encode.base64(val: any, trimTrailEq?: boolean): string`
Returns a base64 representation of the given string or Buffer.
@param – `val` – value to base64-encode.
@param – `trimTrailEq` – whether to trim any trailing `=`.

`C.Encode.gzip`
(method) `Encode.gzip(value: string, encoding?: string): any`
Gzip, and optionally base64-encode, the supplied value.
@param – `value` – the value to gzip.
@param – `encoding` – 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.

`C.Encode.hex`
(method) `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`.
@param – `val` – value to convert to hex.

`C.Encode.uri`
(method) `Encode.uri(val: string): string`
Returns the URI-encoded representation of the given string.
@param – `val` – value to uri encode.

C.env – Environment

`C.env`
(property) `env: {[key: string]: string;}`
Returns an object containing Cribl Edge's environment variables.

C.Lookup – Inline Lookup Functions

`C.Lookup` – Exact Lookup
(property) `Lookup: (file: string, primaryKey?: string, otherFields?: string[], ignoreCase?: boolean) => InlineLookup`
Returns an instance of a lookup to use inline.

Example invocation, where `host` is the name of the primary key field:

```
C.Lookup('lookup_name.csv', 'IP_field_name_in_lookup_file').match(host)
```

Expanded example, where 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_string_to_match',
'field_in_csv_to_output')
```

Example that returns a boolean – is `someValue` there or not?:

```
C.Lookup('file','primaryKeyCol',additionalCols).match('someValue')
```

Example that returns a string – the `fieldToReturn` column from the matched lookup row:

```
C.Lookup('file','primaryKeyCol',additionalCols).match('someValue','fieldToReturn')
```

Example that returns an array – the specified `fieldToReturn1` and `fieldToReturn2` columns from the matched lookup row:

```
C.Lookup('file','primaryKeyCol').match('someValue',['fieldToReturn1', 'fieldToReturn2'])
```

Example that returns an array – all columns from the matched lookup row:

```
C.Lookup('file','primaryKeyCol').match('someValue',[])
```

> `C.Lookup` can load lookup files of up to 10 MB.
>
> All inputs to Lookup functions' `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

(property) `LookupCIDR: (file: string, primaryKey?: string, otherFields?: string[]) => InlineLookup`

Returns an instance of a CIDR lookup to use inline.

`C.LookupIgnoreCase` – Case-insensitive Lookup

(property) `LookupIgnoreCase: (file: string, primaryKey?: string, otherFields?: string[]) => InlineLookup`

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
(property) `LookupRegex: (file: string, primaryKey?: string, otherFields?: string[]) => InlineLookup`
Returns an instance of a Regex lookup to use inline.

(method) `InlineLookup.match(value: string, fieldToReturn?: string): any`
@param – `value` – the value to look up.
@param – `fieldToReturn` – name of the lookup file > field to return.

E.g., `C.Lookup('lookup-exact.csv', 'foo').match('abc', 'bar')`
Return the value of field **bar** in the lookup table if field **foo** matches `abc`.

Example 1: `C.LookupCIDR('lookup-cidr.csv', 'foo').match('192.168.1.1', 'bar')`
Return the value of field **bar** in the lookup table if the CIDR range in **foo** includes `192.168.1.1`.

Example 2: `C.LookupCIDR('lookup-cidr.csv', 'cidr').match(hostIP, 'location')`
Return the value of column **location** in the lookup table if the **location** includes `hostIP`.

Example 3: `C.LookupRegex('lookup-regex.csv', 'foo').match('manchester', 'bar')`
Return the value of field **bar** in the lookup table if the regex in **foo** matches the string `manchester`.

> With `C.LookupRegex`, ensure that your lookup file contains no empty lines – not even at the bottom. Any empty row will cause `C.LookupRegex().match()` to always return `true`.

C.Mask – Data Masking Functions

`C.Mask.CC` (method) `Mask.CC(value: string, unmasked?: number, maskChar?: string): string`
Check whether a value could be a valid credit card number, and mask a subset of the value. By default, all digits except the last 4 will be replaced with `X`.
@param – `value` – a string whose digits should be masked, **if and only if** it could be a valid credit card number.
@param – `unmasked` – number of digits to leave unmasked: positive for left, negative for right, `0` for none.
@param – `maskChar` – a string/char to replace a digit with.

`C.Mask.IMEI` (method) `Mask.IMEI(value: string, unmasked?: number, maskChar?: string): string`

Check whether a value could be a valid IMEI number, and mask a subset of the value. By default, all digits except the last 4 will be replaced with `X`.
@param – `value` – a string whose digits should be masked, **if and only if** it could be a valid IMEI number.
@param – `unmasked` – number of digits to leave unmasked: positive for left, negative for right, `0` for none.
@param – `maskChar` – a string/char to replace a digit with.

`C.Mask.isCC`
(method) `Mask.isCC(value: string): boolean`
Checks whether the given value could be a valid credit card number, by computing the string's Lunh's checksum modulo 10 == `0`.
@param – `value` – a string to check for being a valid credit card number.

`C.Mask.isIMEI`
(method) `Mask.isIMEI(value: string): boolean`
Checks whether the given value could be a valid IMEI number, by computing the string's Lunh's checksum modulo 10 == `0`.
@param – `value` – a string to check for being a valid IMEI number

`C.Mask.luhn`
(method) `Mask.luhn(value: string, unmasked?: number, maskChar?: string): string`
Check that value Lunh's checksum mod 10 is `0`, and mask a subset of the value. By default, all digits except the last 4 will be replaced with `X`. If the value's Lunh's checksum mod 10 is not `0`, then the value is returned unmodified.
@param – `value` – a string whose digits should be masked, **if and only if** the value's Lunh's checksum mod 10 is `0`.
@param – `unmasked` – number of digits to leave unmasked: positive for left, negative for right, `0` for none.
@param – `maskChar` – a string/char to replace a digit with.

`C.Mask.LUHN_SUB` (property) `Mask.LUHN_SUB: any`

`C.Mask.luhnChecksum`
(method) `Mask.luhnChecksum(value: string, mod?: number): 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.
@param – `value` – a string whose digits you want to perform the Lunh checksum on.
@param – `mod` – return checksum modulo this number. If `0`, skip modulo. Default is `10`.

`C.Mask.md5`
(method) `Mask.md5(value: string, len?: string | number): string`
Generate MD5 hash of a given value.
@param – `value` – compute the hash of this.

@param – `len` – 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.

`C.Mask.random`
(method) `Mask.random(len?: string | number): string`
Generates a random alphanumeric string.
@param – `len` – a number indicating the length of the result; or, if a string, use its length.

`C.Mask.REDACTED`
(property) `Mask.REDACTED: string`
The literal `'REDACTED'`.

`C.Mask.repeat`
(method) `Mask.repeat(len?: string | number, char?: string): string`
Generates a repeating char/string pattern, e.g., `XXXX`.
@param – `len` – a number indicating the length of the result; or, if a string, use its length.
@param – `char` – pattern to repeat `len` times.

`C.Mask.sha1` (method) `Mask.sha1(value: string, len?: string | number): string` Generate SHA1 hash of given value.
@param – `value` - compute the hash of this.
@param – `len` - 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

C.Misc – Miscellaneous Utility Functions

`C.Misc.zip()`
(method) `Misc.zip(keys: string[], values: any[], dest?: any): any`
Set the given keys to the corresponding values on the given `dest` object. If `dest` is not provided, a new object will be constructed.
@param – `keys` – field names corresponding to keys.
@param – `values` – values corresponding to values.
@param – `dest` – object on which to set field values.
@returns – object on which the fields were set.

E.g., `people = C.Misc.zip(titles, names)`
Sample data: `titles=['ceo', 'svp', 'vp']`, `names=['foo', 'bar', 'baz']`
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.Net – Network Functions

`C.Net.cidrMatch()`
(method) `Net.cidrMatch(cidrIpRange: string, ipAddress: string): boolean`
Determines whether the supplied IPv4 `ipAddress` is inside the range of addresses identified by `cidrIpRange`. For example: `C.Net.cidrMatch ('10.0.0.0/24', '10.0.0.100')` returns `true`.
@param – `cidrIpRange` – IPv4 address range in CIDR format. E.g., `10.0.0.0/24`.
@param – `ipAddress` – The IPv4 IP address to test for inclusion in `cidrIpRange`.

`C.Net.isIpV4()`
(method) `Net.isIpV4(value: string): boolean`
Determine if the value supplied is an IPv4 address. (Available in Cribl Edge v.3.1.2+.)
@param – `value` – the IP address to test.

`C.Net.isIpV6()`
(method) `Net.isIpV6(value: string): boolean`
Determine if the value supplied is an IPv6 address. (Available in Cribl Edge v.3.1.2+.)
@param – `value` – the IP address to test.

`C.Net.isIpV4AllInterfaces()`
(method) `Net.isIpV4AllInterfaces(value: string): boolean`
Determine if the value supplied is the IPv4 all-interfaces address, typically used to bind to all IPv4 interfaces – i.e., '0.0.0.0'. (Available in Cribl Edge v.3.1.2+.) @param – `value` – the IP address to test.

`C.Net.isIpV6AllInterfaces()`
(method) `Net.isIpV6AllInterfaces(value: string): boolean`
Determine if 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'. (Available in Cribl Edge v.3.1.2+.)
@param – `value` – the IP address to test.

`C.Net.ipv6Normalize()`
(method) `Net.ipv6Normalize(address: string): string`
Normalize an IPV6 address based on RFC draft-ietf-6man-text-addr-representation-04.
@param – `address` – the IPV6 address to normalize.

`C.Net.isPrivate()`
(method) `Net.isPrivate(address: string): string`
Determine whether the supplied IPv4 address is in the range of private addresses per RFC1918.
@param – `address` – address to test.

C.os – System Functions

`C.confVersion`
Returns Cribl Edge config version.

`C.os.hostname()`
Returns hostname of the system running this Cribl Edge instance.

C.Schema – Schema Functions

`C.Schema()`
(property) `Schema: (id: string) => SchemaValidator`
(method) `SchemaValidator.validate(data: any): boolean`
Validates the given object against the schema.
@param – `data` – object to be validated.
@returns – `true` when schema is valid; otherwise, `false`.
Example: `C.Schema('schema1').validate(myField)` will validate if `myField` object conforms to `schema1`.

See Schema Library for more details.

C.Secret – Secrets-Management Functions

`C.Secret()`
(method) `Secret: (id: string, type?: string): ISecret`
(method) `Secret(id: string, type: 'keypair') => IPairSecret`
(method) `Secret(id: string, type: 'text') => ITextSecret`
(method) `Secret(id: string, type: 'credentials') => ICredentialsSecret`
Returns a secret matching the specified ID.
@param `id` – ID of the secret.
@param `type` – optional type of the secret.

Examples:

- `C.Secret('victorias', 'text')` will return a text secret with ID `victorias` (or with undefined, if no such secret exists).
- `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.Text – Text Functions

`C.Text.entropy()`
(method) `Text.entropy(bytes: any): number`
Computes the Shannon entropy of the given buffer or string.
@param – `bytes` – value to undergo Shannon entropy computation.
@returns – the entropy value; or `-1` in case of an error.

`C.Text.hashCode()`
(method) `Text.hashCode(val: string | Buffer | number): number`
Computes hashcode (djb2) of the given value.
@param – `val` - value to be hashed.
@returns – hashcode value.

`C.Text.isASCII()`
(method) `Text.isASCII(bytes: any): boolean`
Checks whether all bytes or chars are in the ASCII printable range.
@param – `bytes` – value to check for character range.
@returns – `true` if all chars/bytes are within ASCII printable range; otherwise, `false`.

`C.Text.isUTF8()`
(method) `Text.isUTF8(bytes: any): boolean`
Checks whether the given Buffer contains valid UTF8.
@param – `bytes` – bytes to check.
@returns – `true` if bytes are UTF8; otherwise, `false`.

`C.Text.parseWinEvent`
(method) `C.Text.parseWinEvent(xml: string, nonValues?: string[]): any`
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.
@param – `xml` – an XML string; or an event field containing the XML.
@param – `nonValues` – 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.
@returns – an object representing the parsed Windows Event; or `undefined` if the input could not be parsed.

`C.Text.parseXml`
(method) `C.Text.parseXml(xml:string, keepAttr?:boolean, keepMetadata?:boolean, nonValues?:string[]): any`
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.
@param – `xml` – XML string, or an event field containing the XML.
@param – `keepAttr` – whether or not to include attributes in the returned object. Defaults to `true`.
@param – `keepMetadata` – 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.)
@param – `nonValues` – 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.
@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.

`C.Text.relativeEntropy()`
(method) `Text.relativeEntropy(bytes: any, modelName?: string): number`
Computes the relative entropy of the given buffer or string.
@param – `bytes` – Value whose relative entropy to compute.
@param – `modelName` – 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.
@returns – The relative entropy value, or `-1` in case of an error.

> 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.

C.Time – Time Functions

`C.Time.adjustTZ()`
(method) `Time.adjustTZ(epochTime: number, tzTo: string, tzFrom?: string): number`
Adjust a timestamp from one timezone to another.
@param – `epochTime` – UNIX epoch time.
@param – `tzTo` – timezone to adjust to.
@param – `tzFrom` – optional timezone of the timestamp.
@returns – the adjusted timestamp, in UNIX epoch time (ms).

> Cribl relies on this timezone-support package, and on this list of TZ Database Time Zones.

`C.Time.clamp()`
(method) `Time.clamp(date, earliest, latest, defaultDate?): number`
Constrains an event's parsed timestamp to realistic earliest and latest boundaries.
@param – `date` – Timestamp originally parsed from event, in UNIX epoch time (ms) or JavaScript Date format.
@param – `earliest` – earliest allowable timestamp, in UNIX epoch time (ms) or JS Date format.
@param – `latest` – latest allowable timestamp, in UNIX epoch time (ms) or JS Date format.
@param – `defaultDate` – optional default date, in UNIX epoch time (ms) or JS Date format, to substitute for values outside the `earliest` or `latest` boundaries.

`C.Time.strftime()`
(method) `Time.strftime(date: number | Date, format: string, utc?: boolean): string`
Format a Date object or number as a time string, using strftime specifier.
@param – `date` – Date object or number (seconds since epoch) to format.
@param – `format` – specifier to use to format the date.
@param – `utc` – whether to output the time in UTC, rather than in local timezone.
@returns – representation of the given date.

`C.Time.strptime()`
(method) `Time.strptime(str: string, format: string, utc?: boolean, strict?: boolean): Date`
Extract time from a string using strptime specifier.
@param – `str` – string to parse to a timestamp (see strict flag).
@param - `format` – `strptime` specifier.
@param – `utc` – whether to interpret times as UTC, rather than as local time.
@param – `strict` – whether to return `null` if there are any extra characters after timestamp.
@returns – a parsed Date object, if successful; otherwise, `null` if the specifier did not match.

`C.Time.timestampFinder()`
(method) `Time.timestampFinder(utc?: boolean).find(<source-field>): AutoTimeParser`
Extract time from the specified `<source-field>`, using the same algorithm as the Auto Timestamp Function and the Event Breaker Function.
@param – `utc` – whether to output the time in UTC, rather than in local timezone.
@param – `<source-field>` – the field in which to search for the time.
@returns – representation of the extracted time; truncates timestamps to three-digit (milliseconds) resolution, omitting trailing zeros.

C.vars – Global Variables

See Global Variables Library for more details.

C.version – Cribl Edge Version

(property) `version: string`
Cribl Edge Version.

14.5. 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`
	`false`
	`null`
	`undefined`

TRUTHY	FALSY
`3.14`	`0`
`"foo"`	`NaN`
`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*`.

14.6. Environment Variables

This is a consolidated list of environment variables available to configure Cribl Edge instances.

Distributed Deployment

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 Formatting Notes below.
`CRIBL_DIST_MODE`	`worker` or `master`. Defaults to `worker`, if (and only if) `CRIBL_DIST_MASTER_URL` is present.
`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_NOAUTH`	Disables authentication. Careful here!!
`CRIBL_TMP_DIR`	Defines the root of a temporary directory. See Formatting 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 folders in the specified directory. If that directory already contains folders with those names, they will be overwritten.
`CRIBL_DIST_WORKER_PROXY`	Communicate to the Leader Node via a SOCKS proxy.See Formatting Notes below.
`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. This variable is not related to `CRIBL_BOOTSTRAP`.
`CRIBL_USERNAME`	Used to log in or out of Cribl.
`CRIBL_PASSWORD`	Used to log in or out of Cribl.

NAME	PURPOSE
`CRIBL_HOST`	The Host URL for authentication. Example: `CRIBL_HOST=<url> CRIBL_USERNAME=<username> CRIBL_PASSWORD=<password> $CRIBL_HOME/bin/cribl auth login`

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>
```

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 assign (Stream, Edge) the Worker 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_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

Use the format `<socks4|socks5>://<username>:<password>@<host>:<port>`. Only `<host>:<port>` are required.

The default protocol is `socks5://`, but you can specify `socks4://proxyhost:port` if needed.

To authenticate on a SOCKS4 proxy with username and password, use this format: `username:password@proxyhost:port`. The `proxyhost` can be a `hostname`, `ip4`, or `ip6`.

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 a Second Leader Node

You can configure a second Leader Node via the following environment variables.

NAME	PURPOSE
`CRIBL_DIST_MASTER_RESILIENCY=failover`	Sets the Leader's `Resiliency` to `Failover` mode.
`CRIBL_DIST_MASTER_FAILOVER_VOLUME=/tmp/shared`	Sets the location of the NFS directory to support Leader failover.
`CRIBL_DIST_MASTER_FAILOVER_MISSED_HB_LIMIT`	Determines how many Lease refresh periods elapse before the 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`.
`CRIBL_INSTANCE_HOME`	In `Failover` mode, this variable points to the Leader Node's `root` directory, as opposed to the shared volume. It is used to access `$CRIBL_INSTANCE_HOME/local/_system/instance.yml`. Outside of `Failover` mode, it's the same value as `CRIBL_CONF_DIR`.

GitOps

Cribl Edge provides the following environment variables to facilitate GitOps.

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`	One of: `push` to enable the GitOps push workflow, or `none` to disable GitOps.
`CRIBL_GIT_SSH_KEY`	Content of the SSH key used to access git remote.
`CRIBL_GIT_STRICT_HOST_KEY_CHECKING`	Boolean flag sets whether to check the host key strictly.
`CRIBL_INTERACTIVE`	Controls whether git commands called by Cribl CLI at startup are interactive.

Internal Environment Variables

Cribl Edge uses the following variables internally.

NAME	PURPOSE
`CRIBL_AUTO_PORTS`	When set to `true`, allows the Cribl process to listen to the first open port, if the designated API port is taken.
`CRIBL_EDGE`	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`	Location of the host OS filesystem when mounting in a container. Defaults to `/hostfs`.
`CRIBL_WORKER_ID`	Passed to Worker processes.
`CRIBL_GROUP_ID`	Passed to ConfigHelper processes to identify Fleets.

NAME	PURPOSE
CRIBL_K8S_FOOTGUN	Set to `true` to enable resource-intensive, potentially risky modes of the Kubernetes Metrics and Kubernetes Logs Sources.
CRIBL_K8S_POD	Sets the name of the Kubernetes Pod in which Cribl Edge is deployed.
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.
CRIBL_ROLE	Controls the behavior of a Cribl subprocess, e.g., `LEADER`, `WORKER`, `CONFIG_HELPER`.
CRIBL_SERVICE	Set to 1 when using **systemd** to start Cribl at boot time.
CRIBL_SERVICE_NAME	Set to `cribl` when using **systemd** to start Cribl at boot time.
CRIBL_SERVICEACCOUNT_PATH	Path to the `ServiceAccount` to use to query the Kubernetes API. Defaults to `/var/run/secrets/kubernetes.io/serviceaccount`.
CRIBL_SPOOL_DIR	Path to write metrics to disk. Defaults to `/$CRIBL_HOME/state/spoolin/source_type/source_id` and adjusted for `$CRIBL_VOLUME_DIR` if set.

14.7. 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

In the Source 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.

The meaning of All and Disabled are self-evident. Basic and Custom have different meanings depending on event type and will be described under each section below.

The tables outline the metrics emitted for each mode (Basic or Custom) and where applicable, the dimensions (to indicate where the metrics are coming from).

System

With System Metrics enabled, Cribl Edge captures CPU load averages, uptime, and count. The **Custom** option allows you to include process metrics that reflect the numbers of processes in various states.

Metrics for the overall system include the following:

NAME	DESCRIPTION	TYPE	DIMENSIONS	MODE
`node_uname_info`	Labeled system information as provided by the uname system	Counter	`release`, `sysname`,	Basic

NAME	DESCRIPTION	TYPE	DIMENSIONS	MODE
	call.		`version`	
`node_cpu_count`	The number of CPU cores.	Gauge	`release`, `sysname`, `version`	Basic
`node_uptime_seconds`	System uptime in seconds.	Counter	N/A	Basic
`node_boot_time_seconds`	Node boot time in Unix time.	Counter	N/A	Basic
`node_time_seconds`	System time in seconds.	Counter	N/A	Basic
`node_load1`	1m load average.	Gauge	N/A	Basic
`node_load5`	5m load average.	Gauge	N/A	Basic
`node_load15`	15m load average.	Gauge	N/A	Basic
`node_open_fds`	Open file descriptors	Counter	N/A	Basic
`node_processes_state_all`	Total number of processes in different states.	Gauge	`state`	Basic
`node_processes_threads`	Allocated threads in system.	Gauge	`state`	Basic
`node_procs_blocked`	Number of processes blocked waiting for I/O to complete.	Gauge	`state`	Basic
`node_procs_running`	Number of processes in runnable state.	Gauge	`state`	Basic
`node_processes_state`	Number of processes in each state.	Gauge	`state`	Custom: `Process metrics`

CPU

Cribl Edge captures active, user, system, idle, and `iowait` percentages over all CPUs, with options to add per-CPU metrics and raw time counters for each state.

Metrics for CPUs include the following:

NAME	DESCRIPTION	TYPE	DIMENSIONS	MODE
`node_cpu_percent_active_all`	Percent all the CPUs spent in activity.	Gauge	N/A	Basic
`node_cpu_seconds_active_all_total`	Seconds all the CPUs spent in activity (excluding idle and wait).	Counter	N/A	Custom: `CPU time metrics`
`node_cpu_seconds_active_total`	Seconds each CPU spent in activity (excluding idle and wait).	Counter	`cpu`	Custom: `CPU time metrics`
`node_cpu_seconds_all_total`	Seconds for all CPUs usage.	Counter	`mode`	Custom: `CPU time metrics`
`node_cpu_seconds_total`	Seconds for each CPU's usage.	Counter	`cpu`, `mode`	Custom: `CPU time metrics`
`node_cpu_percent_active`	Percent each CPU spent in activity.	Gauge	`cpu`	Custom: `Per CPU` or `Detailed` metrics
`node_cpu_percent_all`	Percent CPU usage for all.	Gauge	`mode`,`user`	Custom: `Detailed` metrics
`node_cpu_percent`	Percent CPU usage for each CPU.	Gauge	`cpu`,`mode`, `user`	Custom: `Per CPU` or `Detailed` metrics

- The `Per CPU` option adds metrics with the `cpu` dimension.
- The `Detailed metrics` option adds metrics with `mode` dimension set to: `irq`, `softirq`, `steal`, `guest`, `guest_nice`, and `nice`.

Memory

With System Metrics enabled, Cribl Edge captures memory metrics including total, used, available, `swap_free`, and `swap_total`, with the option to toggle all memory states.

Metrics for memory include the following:

NAME	DESCRIPTION	TYPE	DIMENSIONS	MODE
node_memory_MemTotal_bytes	Memory information field MemTotal_bytes.	Gauge	N/A	Basic
node_memory_Used_bytes	Used memory in bytes.	Gauge	N/A	Basic
node_memory_Used_percent	Percent used memory.	Gauge	N/A	Basic
node_memory_MemAvailable_bytes	Memory information field MemAvailable_bytes.	Gauge	N/A	Basic
node_memory_MemAvailable_percent	Percent memory available.	Gauge	N/A	Basic
node_memory_SwapFree_bytes	Memory information field SwapFree_bytes.	Gauge	N/A	Basic
node_memory_SwapTotal_bytes	Memory information field SwapTotal_bytes.	Gauge	N/A	Basic
node_vmstat_oom_kill	/proc/vmstat information field oom_kill.	Gauge	N/A	Basic
node_vmstat_pgfault	/proc/vmstat information field pgfault.	Gauge	N/A	Basic
node_vmstat_pgmajfault	/proc/vmstat information field pgmajfault.	Gauge	N/A	Basic
node_vmstat_pgpgin	/proc/vmstat information field pgpgin.	Gauge	N/A	Basic
node_vmstat_pgpgout	/proc/vmstat information field pgpgout.	Gauge	N/A	Basic
node_vmstat_pswpin	/proc/vmstat information field 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

NAME	DESCRIPTION	TYPE	DIMENSIONS	MODE
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	MODE
node_network_receive_bytes_all_total	Network device statistic receive_bytes.	Counter	N/A	Basic
node_network_receive_errs_all_total	Network device statistic receive_errs.	Counter	N/A	Basic
node_network_receive_packets_all_total	Network device statistic receive_packets.	Counter	N/A	Basic
node_network_transmit_bytes_all_total	Network device statistic transmit_bytes.	Counter	N/A	Basic
node_network_transmit_errs_all_total	Network device statistic transmit_errs.	Counter	N/A	Basic
node_network_transmit_packets_all_total	Network device statistic transmit_packets.	Counter	N/A	Basic
node_socket_tcp_established_total	TCP established connections.	Counter	N/A	Basic

NAME	DESCRIPTION	TYPE	DIMENSIONS	MODE
node_network_receive_bytes_total	Network device statistic receive_bytes per interface.	Counter	device	Custom: Per Interface
node_network_receive_errs_total	Network device statistic receive_errs per interface.	Counter	device	Custom: Per Interface
node_network_receive_packets_total	Network device statistic receive_packets per interface.	Counter	device	Custom: Per Interface
node_network_transmit_bytes_total	Network device statistic transmit_bytes per interface.	Counter	device	Custom: Per Interface
node_network_transmit_errs_total	Network device statistic transmit_errs per interface.	Counter	device	Custom: Per Interface
node_network_transmit_packets_total	Network device statistic transmit_packets per interface.	Counter	device	Custom: Per Interface
node_network_receive_drop_all_total	Network device statistic receive_drop.	Counter	N/A	Custom: Detailed Metrics
node_network_receive_drop_total	Network device statistic receive_drop per interface.	Counter	device	Custom: Detailed Metrics
node_network_transmit_drop_all_total	Network device statistic transmit_drop.	Counter	N/A	Custom: Detailed Metrics
node_network_transmit_drop_total	Network device statistic transmit_drop per interface.	Counter	device	Custom: Detailed Metrics
node_socket_tcp_syn_sent_total	TCP sent packets total.	Counter	N/A	Custom: Detailed Metrics
node_socket_tcp_syn_recv_total	TCP received packets total.	Counter	N/A	Custom: Detailed Metrics

NAME	DESCRIPTION	TYPE	DIMENSIONS	MODE
`node_socket_tcp_fin_wait1_total`	Total connections waiting for termination request from remote TCP.	Counter	N/A	Custom: `Detailed Metrics`
`node_socket_tcp_fin_wait2_total`	Active TCP connections to be shut down.	Counter	N/A	Custom: `Detailed Metrics`
`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	Custom: `Detailed Metrics`
`node_socket_tcp_close_total`	Total TCP sockets with closed connections.	Counter	N/A	Custom: `Detailed Metrics`
`node_socket_tcp_last_ack_total`	Total TCP sockets in state before the TCP connection is closed.	Counter	N/A	Custom: `Detailed Metrics`
`node_socket_tcp_listen_total`	Total TCP sockets waiting for a connection request from any remote TCP/port.	Counter	N/A	Custom: `Detailed Metrics`
`node_socket_tcp_closing_total`	Total TCP sockets waiting for connection termination request acknowledgement.	Counter	N/A	Custom: `Detailed Metrics`
`node_socket_tcp_none_total`	Number of TCP sockets with no connections.	Counter	N/A	Custom: `Detailed Metrics`
`node_socket_udp_total`	Number of UDP sockets in use.	Counter	N/A	Custom: `Detailed Metrics`

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	MODE
`node_disk_reads_completed_all_total`	Total number of reads completed successfully.	Counter	N/A	Basic
`node_disk_read_bytes_all_total`	Total number of bytes read successfully.	Counter	N/A	Basic
`node_disk_writes_completed_all_total`	Network device statistic `receive_packets`.	Counter	N/A	Basic
`node_disk_written_bytes_all_total`	Total number of bytes written successfully.	Counter	N/A	Basic
`node_filesystem_size_bytes_all`	Filesystem size in bytes.	Gauge	N/A	Basic
`node_filesystem_avail_bytes_all`	Filesystem space available to non-root users in bytes.	Gauge	N/A	Basic
`node_filesystem_used_bytes_all`	Filesystem used space in bytes.	Gauge	N/A	Basic
`node_filesystem_used_percent_all`	Percent filesystem used space.	Gauge	N/A	Basic
`node_filesystem_files_free_all`	Filesystem free file nodes.	Gauge	N/A	Basic
`node_filesystem_files_used_all`	Filesystem total used file nodes.	Gauge	N/A	Basic
`node_filesystem_files_used_percent_all`	Percent Filesystem used in all disks.	Gauge	N/A	Basic
`node_filesystem_size_bytes_all`	Filesystem size in bytes.	Gauge	N/A	Basic
`node_filesystem_used_bytes`	Filesystem used space in bytes per disk.	Gauge	`device`, `fstype`, `mountpoint`	Custom: `Per device metrics`

NAME	DESCRIPTION	TYPE	DIMENSIONS	MODE
				or `Detailed metrics`
`node_filesystem_used_percent`	Percent Filesystem used per disk.	Gauge	`device`, `fstype`, `mountpoint`	Custom: `Per device metrics` or `Detailed metrics`
`node_disk_reads_completed_total`	Total number of reads completed successfully per disk.	Counter	`device`	Custom: `Per device metrics`
`node_disk_read_bytes_total`	Total number of bytes read successfully per disk.	Counter	`device`, `fstype`, `mountpoint`	Custom: `Per device metrics`
`node_disk_writes_completed_total`	Network device statistic `receive_packets` per disk.	Counter	`device`, `fstype`, `mountpoint`	Custom: `Per device metrics`
`node_disk_written_bytes_total`	Total number of bytes written successfully per disk.	Counter	`device`, `fstype`, `mountpoint`	Custom: `Per device metrics`
`node_disk_discards_completed_all_total`	Total number of discards completed successfully.	Counter	N/A	Custom: `Detailed Metrics`
`node_disk_discards_completed_total`	Total number of discards completed successfully per disk.	Counter	`device`, `fstype`, `mountpoint`	Custom: `Per device metrics` or `Detailed metrics`
`node_disk_discards_merged_all_total`	Total number of discards merged.	Counter	N/A	Custom: `Detailed metrics`
`node_disk_read_time_seconds_total`	Total number of seconds spent by all reads per disk.	Counter	`device`, `fstype`,	Custom: `Detailed metrics`

NAME	DESCRIPTION	TYPE	mountpoint	MODE
node_disk_write_time_seconds_all_total	Total number of seconds spent by all writes.	Counter	N/A	Custom: Detailed Metrics
node_disk_read_time_seconds_all_total	Total number of seconds spent by all reads.	Counter	N/A	Custom: Detailed Metrics
node_disk_write_time_seconds_total	Total number of seconds spent by all writes per disk.	Counter	device, fstype, mountpoint	Custom: Per device metrics or Detailed metrics
node_disk_reads_merged_all_total	Total number of reads merged.	Counter	N/A	Custom: Detailed Metrics
node_disk_writes_merged_all_total	Total number of writes merged.	Counter	N/A	Custom: Detailed metrics
node_disk_reads_merged_total	Total number of reads merged per disk.	Counter	device, fstype, mountpoint	Custom: Per device metrics or Detailed metrics
node_disk_writes_merged_total	Total number of writes merged per disk.	Counter	device, fstype, mountpoint	Custom: Per device metrics or Detailed metrics
node_disk_discards_merged_total	Total number of discards merged per disk.	Counter	device, fstype, mountpoint	Custom: Per device metrics or Detailed metrics
node_disk_io_time_seconds_all_total	Total seconds spent doing I/Os.	Counter	N/A	Custom: Detailed Metrics

NAME	DESCRIPTION	TYPE	DIMENSIONS	MODE
node_disk_io_time_seconds_total	Total seconds spent doing I/Os per disk.	Counter	device, fstype, mountpoint	Custom: Per device metrics or Detailed metrics
node_disk_io_now_all	The number of I/Os currently in progress.	Gauge	N/A	Custom: Detailed Metrics
node_disk_io_now	The number of I/Os currently in progress per disk.	Counter	device, fstype, mountpoint	Custom: Per device metrics or Detailed metrics
node_disk_io_time_weighted_seconds_all_total	Weighted number of seconds spent doing I/Os per device.	Counter	device, fstype, mountpoint	Custom: Per device metrics or Detailed metrics
node_filesystem_size_bytes	Filesystem size in bytes per disk.	Gauge	device, fstype, mountpoint	Custom: Per device metrics or Detailed metrics
node_filesystem_avail_bytes	Filesystem space available to non-root users in bytes per disk.	Gauge	device, fstype, mountpoint	Custom: Per device metrics or Detailed metrics
node_filesystem_files_free	Filesystem free file nodes per disk.	Gauge	device, fstype, mountpoint	Custom: Per device metrics or Detailed metrics

NAME	DESCRIPTION	TYPE	DIMENSIONS	MODE
`node_filesystem_files_used_percent`	Percent Filesystem used per disk.	Gauge	`device`, `fstype`, `mountpoint`	Custom: `Per device metrics` or `Detailed metrics`
`node_filesystem_size_bytes`	Filesystem size in bytes per disk.	Gauge	`device`, `fstype`, `mountpoint`	Custom: `Per device metrics` or `Detailed metrics`

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	DIMENSIONS
`container_start_time_seconds`	UNIX time (seconds since the epoch) when the container was started.	Counter	N/A
`container_finish_time_seconds`	UNIX time (seconds since the epoch) when the container was stopped. Only for non-running containers.	Counter	N/A
`container_fs_reads_bytes_all_total`	Total bytes read for all disk devices.	Counter	N/A
`container_memory_usage_percent`	Percent of available memory being used.	Gauge	N/A
`container_network_receive_bytes_all_total`	Total bytes received for all network interfaces.	Counter	N/A
`container_network_receive_errors_all_total`	Total number of errors received for all network interfaces.	Counter	N/A

NAME	DESCRIPTION	TYPE	DIMENSIONS
container_network_receive_packets_all_total	Total number of packets received for all network interfaces.	Counter	N/A
container_network_transmit_bytes_all_total	Total bytes transmitted for all network interfaces.	Counter	N/A
container_network_transmit_errors_all_total	Total number of errors transmitted for all network interfaces.	Counter	N/A
container_network_transmit_packets_all_total	Total number of packets transmitted for all network interfaces.	Counter	N/A
container_memory_total_bytes	Total number of memory bytes available.	Counter	N/A
container_cpu_user_seconds_total	Number of seconds the container has been on the CPU running user code.	Counter	cpu
container_cpu_system_seconds_total	Number of seconds the container has been on the CPU running kernel code.	Counter	cpu
container_fs_reads_bytes_total	Total bytes read per device	Counter	device
container_fs_writes_bytes_all_total	Total bytes written for all disk devices	Counter	N/A
container_fs_writes_bytes_total	Total bytes written per device	Counter	device

NAME	DESCRIPTION	TYPE	DIMENSIONS
container_fs_reads_all_total	Total number of read operations for all disk devices.	Counter	N/A
container_fs_writes_total	Total number write operations per device.	Counter	device
container_memory_mapped_file	Total bytes writted for all disk devices.	Counter	N/A
container_memory_max_usage_bytes	Highest seen value of the container_memory_usage_bytes metric.	Counter	N/A
container_memory_pgin	Total number of memory page-in events.	Counter	N/A
container_mem.pgpgout	Total number of memory pages paged out.	Counter	N/A
container_memory_pgfault	Total number of major page faults.	Counter	N/A
container_memory_pgmajfault	Total number of minor page faults.	Counter	N/A
container_memory_usage_bytes	Number of memory bytes used.	Counter	N/A
container_network_receive_dropped_all_total	Total number of receives dropped for all network interfaces.	Counter	N/A
container_network_transmit_dropped_all_total	Total number of transmits dropped for all network interfaces.	Counter	N/A

14.8. 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

In the Source 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.

The meaning of **All** and **Disabled** are self-evident. **Basic** and **Custom** have different meanings depending on event type and will be described under each section below.

The tables outline the metrics emitted for each mode (**Basic** or **Custom**) and where applicable, the dimensions (to indicate where the metrics are coming from).

System

With System Metrics enabled, Cribl Edge captures CPU load averages, uptime, and count. The **Custom** option option allows you to include detailed metrics. These are Windows-specific metrics including OS information, system uptime, CPU architecture, etc.

Metrics for the overall system include the following:

NAME	DESCRIPTION	TYPE	DIMENSIONS	MODE
`windows_cs_logical_processors`	Number of installed logical	Gauge	N/A	Basic

NAME	DESCRIPTION	TYPE	DIMENSIONS	MODE
	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	Custom: Detailed
windows_cpu_info	Labeled CPU information.	Gauge	architecture, device_id, description, family, l2_cache_size, l3_cache_size, name	Custom: Detailed
windows_os_paging_limit_bytes	Total number of bytes that can	Gauge	N/A	Custom: Detailed

NAME	DESCRIPTION	TYPE	DIMENSIONS	MODE
	be stored in the operating system paging files.			
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	Custom: Detailed
windows_os_processes_limit	Maximum number of process contexts the operating system can support.	Gauge	N/A	Custom: Detailed
windows_os_process_memory_limit_bytes	Maximum number of bytes of memory that can be allocated to a process.	Gauge	N/A	Custom: Detailed
windows_os_virtual_memory_bytes	Bytes of virtual memory.	Gauge	N/A	Custom: Detailed
windows_system_exception_dispatches_total	Total exceptions dispatched by the system.	Counter	N/A	Custom: Detailed
windows_system_system_calls_total	Total combined calls to Windows NT system service routines by all processes running on the computer.	Counter	N/A	Custom: Detailed
windows_system_system_up_time	Time of last boot of system.	Gauge	N/A	Custom: Detailed

CPU

Basic level captures active, user, system, idle, and `iowait` percentages over all CPUs.

Custom level toggles the following on or off: **Per CPU metrics**, **Detailed metrics** (i.e., metrics for all CPU states), and **CPU time metrics** (i.e., raw, monotonic CPU time counters).

Metrics for CPUs include the following:

NAME	DESCRIPTION	TYPE	DIMENSIONS	MODE
`windows_cpu_percent_active_all`	CPU percent active usage	Gauge	`core`, `mode`	Basic
`windows_cpu_percent_active`	CPU percent active usage per CPU	Gauge	`core`, `mode`	Basic or Custom: `Per CPU` and `CPU time` metrics
`windows_cpu_percent`	CPU percent active usage	Gauge	`core`, `mode` is set to `user`, `idle`, `privileged`, `interrupt`, `dpc`	Basic or Custom: `Per CPU` and `CPU time` metrics
`windows_cpu_parking_status`	Parking Status represents whether a processor is parked or not.	Counter	`core`	Basic or Custom: `Per CPU` and `CPU time` metrics
`windows_cpu_core_frequency_mhz`	Core frequency in megahertz.	Gauge	`core`	Basic or Custom: `Per CPU` and `CPU time` metrics
`windows_cpu_time_all_total`	Sum of all `cpu_time` across all cores.	Gauge	`mode`	Basic or Custom: `CPU time` metrics
`windows_cpu_cstate_seconds_total`	Time spent in low-power idle state.	Counter	`core`, `state`	Custom: `Per CPU` and

NAME	DESCRIPTION	TYPE	DIMENSIONS	MODE
				`Detailed` metrics
`windows_cpu_time_total`	Time that processor spent in different modes (idle, user, system etc.).	Counter	`core`, `mode`	Custom: `Per CPU` and `CPU time` metrics
`windows_cpu_interrupts_total`	Total number of received and serviced hardware interrupts.	Counter	`core`	Custom: `Per CPU` or `Detailed` metrics
`windows_cpu_dpcs_total`	Total number of received and serviced deferred procedure calls (DPCs).	Counter	`core`	Custom: `Per CPU` or `Detailed` metrics
`windows_cpu_clock_interrupts_total`	Total number of received and serviced clock tick interrupts.	Counter	`core`	Custom: `Per CPU` or `Detailed` metrics
`windows_cpu_idle_break_events_total`	Total number of time processor was woken from idle.	Counter	`core`	Custom: `Per CPU` or `Detailed` metrics
`windows_cpu_processor_performance`	Average performance of the processor while it is executing instructions.	Gauge.	`core`	Custom: `Per CPU` and `Detailed` metrics
`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`	Custom: `Per CPU` and `Detailed` metrics

NAME	DESCRIPTION	TYPE	DIMENSIONS	MODE
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 its nominal performance and never idle.	Gauge.	core	Custom: Per CPU and Detailed metrics
windows_cpu_average_idle_time	Processor idle time.	Gauge	mode	Custom: Per CPU and Detailed metrics
windows_cpu_percent_privilege_utility	Amount of work a processor is completing while executing in privileged mode.	Gauge	mode	Custom: Per CPU and Detailed metrics
windows_cpu_interrupts_total_per_sec	Total number of received and serviced hardware interrupts, computed average on a per second interval.	Gauge	mode	Custom: Per CPU and Detailed metrics
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	Custom: Per CPU and Detailed metrics
windows_cpu_clock_interrupts_total_per_sec	Total number of received and serviced clock tick interrupts, computed average on a per second interval.	Gauge	mode	Custom: Per CPU and Detailed metrics

NAME	DESCRIPTION	TYPE	DIMENSIONS	MODE
`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`	Custom: `Per CPU` and `Detailed` metrics
`windows_cpu_percent_all_total`	Core frequency in megahertz.	Gauge	`mode` is set to `dpc`, `idle`, `interrupt`, `privilege`, `user`,`active`	Custom: `CPU time` metrics

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	DIMENSIONS	MODE
`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	Basic
`windows_memory_cache_bytes`	Number of bytes currently being used by the filesystem cache	Gauge	N/A	Basic
`windows_memory_cache_bytes_peak`	Maximum number of `CacheBytes` after the system	Gauge	N/A	Basic

NAME	DESCRIPTION	TYPE	DIMENSIONS	MODE
	was last restarted.			
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	Basic
windows_memory_commit_limit	Bytes of virtual memory that can be committed without having to extend paging files.	Gauge	N/A	Basic
windows_memory_committed_bytes	Bytes of committed virtual memory.	Gauge	N/A	Basic
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	Basic
windows_memory_pool_paged_bytes	Number of bytes in the paged pool.	Gauge	N/A	Basic
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 memory. The paged pool is an area of the system virtual memory used for	Gauge	N/A	Basic

NAME	DESCRIPTION	TYPE	DIMENSIONS	MODE
	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	Custom: Detaile
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	Custom: Detaile
windows_memory_free_system_page_table_entries	Page table entries not being used by the system.	Gauge	N/A	Custom: Detaile
windows_memory_modified_page_list_bytes	Physical memory, in bytes, assigned to the modified page list. This memory contains cached data and code that is not	Gauge	N/A	Custom: Detaile

NAME	DESCRIPTION	TYPE	DIMENSIONS	MODE
	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	Custom: Detailed
windows_memory_swap_page_reads_total	Disk page reads (a single read operation reading several pages is still only counted once).	Counter	N/A	Custom: Detailed
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	Custom: Detailed
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	Custom: Detailed
windows_memory_swap_page_operations_total	Total number of swap page read and writes (PagesPersec).	Counter	N/A	Custom: Detailed

NAME	DESCRIPTION	TYPE	DIMENSIONS	MODE
windows_memory_swap_page_writes_total	Disk page writes (a single write operation writing several pages is still only counted once).	Counter	N/A	Custom: Detaile
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	Custom: Detaile
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	Custom: Detaile
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 that is not actively in use by processes, the system, and the	Gauge	N/A	Custom: Detaile

NAME	DESCRIPTION	TYPE	DIMENSIONS	MODE
	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 repurposed before memory on higher	Gauge	N/A	Custom: `Detaile`

NAME	DESCRIPTION	TYPE	DIMENSIONS	MODE
	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	Custom: Detaile
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	Custom: Detaile
windows_memory_system_code_resident_bytes	The size, in bytes, of the pageable operating system code that is currently	Gauge	N/A	Custom: Detaile

NAME	DESCRIPTION	TYPE	DIMENSIONS	MODE
	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	Custom: Detaile
windows_memory_system_driver_resident_bytes	The size, in bytes, of the	Gauge	N/A	Custom: Detaile

NAME	DESCRIPTION	TYPE	DIMENSIONS	MODE
	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	Custom: `Detaile`

NAME	DESCRIPTION	TYPE	DIMENSIONS	MODE
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	Custom: Detaile
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	Custom: Detaile

NAME	DESCRIPTION	TYPE	DIMENSIONS	MODE
	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	Custom: Detaile
windows_memory_used_percent	Percent memory used.	Gauge	N/A	Custom: Detaile
windows_memory_available_percent	Percent memory available.	Gauge	N/A	Custom: Detaile
windows_memory_cache_faults_per_sec	Rate of cache faults computed per sec.	Gauge	N/A	Custom: Detaile
windows_memory_demand_zero_faults_per_sec	Rate of Zeroed pages required to satisfy faults computed per sec.	Gauge	N/A	Custom: Detaile
windows_memory_page_faults_per_sec	Rate at which faulted pages are handled by the processor computed per sec.	Gauge	N/A	Custom: Detaile
windows_memory_page_reads_per_sec	Disk page reads computed per sec.	Gauge	N/A	Custom: Detaile
windows_memory_pages_input_per_sec	Disk page reads computed per sec.	Gauge	N/A	Custom: Detaile
windows_memory_pages_output_per_sec	Pages written across all page writes computed per sec.	Gauge	N/A	Custom: Detaile

NAME	DESCRIPTION	TYPE	DIMENSIONS	MODE
windows_memory_pages_per_sec	Total number of swap page read and writes computed per sec.	Gauge	N/A	Custom: Detaile
windows_memory_page_writes_per_sec	Disk page writes computed per sec.	Gauge	N/A	Custom: Detaile
windows_memory_transition_faults_sec	Rate at which page faults are resolved computed per sec.	Gauge	N/A	Custom: Detaile
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	Custom: Detaile
windows_memory_write_copies_per_sec	Rate at which the 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	Custom: Detaile

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	DIMENSIONS	MODE
`windows_net_packets_outbound_discarded_total`	Total outbound packets to be discarded even though no errors had been detected to prevent transmission.	Counter	`nic`	Custom `Detai` metrics
`windows_net_packets_outbound_errors_total`	Total packets that could not be transmitted due to errors.	Counter	`nic`	Custom `Detai` metrics
`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`	Custom `Detai` metrics
`windows_net_packets_received_errors_total`	Total packets that could not be received due to errors.	Counter	`nic`	Custom `Detai` metrics
`windows_net_packets_received_unknown_total`	Total packets received by interface that were	Counter	`nic`	Custom `Detai` metrics

NAME	DESCRIPTION	TYPE	DIMENSIONS	MODE
	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	Custom Detai metrics
windows_net_packets_received_unicast_total	Total subnet-unicast packets that are delivered to a higher-layer protocol.	Counter	nic	Custom Detai metrics
windows_net_packets_sent_unicast_total	Total packets requested to be transmitted to subnet-unicast addresses by higher-level protocols.	Counter	nic	Custom Detai metrics
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	Custom Detai metrics

NAME	DESCRIPTION	TYPE	DIMENSIONS	MODE
windows_net_bytes_received_total	Total bytes received by interface.	Counter	nic	Custom Per Inter metrics
wwindows_net_bytes_received_total_per_sec	Total bytes received by interface computed per sec.	Gauge	nic	Custom Per Inter metrics Detai metrics
windows_net_bytes_sent_total	Total bytes transmitted by interface.	Counter	nic	Custom Per Inter metrics
windows_net_bytes_sent_total_per_sec	Total bytes transmitted by interface computed per sec.	Gauge	nic	Custom Per Inter metrics Detai metrics
windows_net_bytes_total	Total bytes received and transmitted by interface.	Counter	nic	Custom Per Inter metrics
windows_net_bytes_total_per_sec	Total bytes received and transmitted by interface per sec.	Gauge	nic	Custom Per Inter metrics Detai metrics
windows_net_packets_received_total_per_sec	Total packets received by interface computed per sec.	Counter	nic	Custom Per Inter metrics Detai metrics
windows_net_bytes_total_per_sec	Total bytes received and transmitted by interface per sec.	Gauge	nic	Custom Per Inter metrics Detai metrics

NAME	DESCRIPTION	TYPE	DIMENSIONS	MODE
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	Custom Per Inter metrics Detai metrics
windows_net_packets_total	Total packets received and transmitted by interface.	Counter	nic	Custom Per Inter metrics
windows_net_packets_total_per_sec	Total packets received and transmitted by interface computed per sec.	Gauge	nic	Custom Per Inter metrics Detai metrics
windows_net_packets_sent_total	Total packets transmitted by interface.	Counter	nic	Custom Per Inter metrics
windows_net_packets_sent_total_per_sec	Total packets transmitted by interface computed per sec.	Gauge	nic	Custom Per Inter metrics Detai metrics
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	Custom Per Inter metrics Detai metrics

NAME	DESCRIPTION	TYPE	DIMENSIONS	MODE
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	Custom Per Inter metrics Detai metrics
windows_net_current_bandwidth_bytes	Estimate of the interface's current bandwidth in bytes per second.	Gauge	nic	Custom Detai metrics
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	Custom Detai and Proto metrics TCPv4 TCPv6
windows_tcp_connections_active_all_total	Number of times TCP connections have made a	Counter	af	Custom Detai and Proto

NAME	DESCRIPTION	TYPE	DIMENSIONS	MODE
	direct transition from the CLOSED state to the SYN-SENT state.			metrics `TCPv4` `TCPv6`
`windows_tcp_connections_established`	Number of TCP connections for which the current state is either ESTABLISHED or CLOSE-WAIT.	Gauge	`af`	Custom `Detai` and `Proto` metrics `TCPv4` `TCPv6`
`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`	Custom `Detai` and `Proto` metrics `TCPv4` `TCPv6`
`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`	Custom `Detai` and `Proto` metrics `TCPv4` `TCPv6`
`windows_tcp_segments_total`	Total segments sent or received using the TCP protocol.	Counter	`af`	Custom `Detai` and `Proto` metrics `TCPv4` `TCPv6`
`windows_tcp_segments_received_all_total`	Total segments received	Counter	`af`	Custom `Detai` and `Proto`

NAME	DESCRIPTION	TYPE	DIMENSIONS	MODE
	using the TCP protocol.			metrics TCPv4 TCPv6
windows_tcp_segments_retransmitted_all_total	Total segments retransmitted using the TCP protocol.	Counter	af	Custom Detai and Proto metrics TCPv4 TCPv6
windows_tcp_segments_sent_all_total	Total segments sent using the TCP protocol.	Counter	af	Custom Detai and Proto metrics TCPv4 TCPv6
windows_tcp_segments_all_total_per_sec	Total segments sent or received using the TCP protocol computed per sec.	Gauge	af	Custom Detai and Proto metrics TCPv4 TCPv6
windows_tcp_segments_received_all_total_per_sec	Total segments received using the TCP protocol computed per sec.	Gauge	af	Custom Detai and Proto metrics TCPv4 TCPv6
windows_tcp_segments_retransmitted_all_total_per_sec	Total segments retransmitted using the TCP protocol computed per sec.	Gauge	af	Custom Detai and Proto metrics TCPv4 TCPv6
windows_tcp_segments_sent_all_total_per_sec	Total segments sent using the TCP protocol	Gauge	af	Custom Detai and Proto metrics

NAME	DESCRIPTION	TYPE	DIMENSIONS	MODE
	computed per sec.			TCPv4 TCPv6
windows_net_datagrams_all_total	Total datagrams sent or received using the UDP protocol.	Counter	af	Custom Detai and Proto metrics UDPv4 UDPv6
windows_net_datagrams_no_port_all_total	Rate of received UDP datagrams for which there was no application at the destination port.	Gauge	af	Custom Detai and Proto metrics UDPv4 UDPv6
windows_net_datagrams_received_all_total	Rate at which UDP datagrams are delivered to UDP users.	Gauge	af	Custom Detai and Proto metrics UDPv4 UDPv6
windows_net_datagrams_received_errors_all_total	Number of received UDP datagrams that could not be delivered excluding errors due to lack of an application at the destination port.	Gauge	af	Custom Detai and Proto metrics UDPv4 UDPv6
windows_net_datagrams_sent_all_total	Total UDP datagrams sent from the entity.	Gauge	af	Custom Detai and Proto metrics UDPv4 UDPv6

NAME	DESCRIPTION	TYPE	DIMENSIONS	MODE
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	Custom Detai and Proto metrics UDPv4 UDPv6
windows_net_datagrams_received_all_total_per_sec	Rate at which UDP datagrams are delivered to UDP users computed per sec	Gauge	af	Custom Detai and Proto metrics UDPv4 UDPv6
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	Custom Detai and Proto metrics UDPv4 UDPv6
windows_net_datagrams_sent_all_total_per_sec	Rate at which UDP datagrams sent from the entity computed per sec.	Gauge	af	Custom Detai and Proto metrics UDPv4 UDPv6

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	DIMENSIONS	MODE
`windows_logical_disk_requests_queued`	Outstanding requests on the disk at the time the performance data is collected	Gauge	`volume`	Basic
`windows_logical_disk_read_bytes_total`	Rate at which bytes are transferred from the disk during read operations.	Counter	`volume`	Basic
`windows_logical_disk_reads_total`	Rate of read operations on the disk.	Counter	`volume`	Basic
`windows_logical_disk_write_bytes_total`	Rate at which bytes are transferred to the disk during write operations.	Counter	`volume`	Basic
`windows_logical_disk_writes_total`	Rate of write operations on the disk.	Counter	`volume`	Basic
`windows_logical_disk_write_latency_seconds_total`	Shows the average	Counter	`volume`	Custom: `Detailed`

NAME	DESCRIPTION	TYPE	DIMENSIONS	MODE
	time, in seconds, of a write operation to the disk.			metrics
windows_logical_disk_read_latency_seconds_total	Shows the average time, in seconds, of a read operation from the disk.	Counter	volume	Custom: Detailed metrics
windows_logical_disk_read_write_latency_seconds_total	Shows the time, in seconds, of the average disk transfer.	Counter	volume	Custom: Detailed metrics
windows_logical_disk_read_seconds_total	Seconds the disk was busy servicing read request.	Counter	volume	Custom: Detailed metrics
windows_logical_disk_idle_seconds_total	Seconds the disk was idle (not servicing read/write requests).	Counter	volume	Custom: Detailed metrics
windows_logical_disk_split_ios_total	Number of I/Os to the disk split into multiple I/Os.	Counter	volume	Custom: Detailed metrics
windows_logical_disk_percent_read_time	Percent rate of read operations on the disk.	Gauge	volume	Custom: Detailed metrics
windows_logical_disk_percent_write_time	Percent write operations on the disk.	Gauge	volume	Custom: Detailed metrics

NAME	DESCRIPTION	TYPE	DIMENSIONS	MODE
windows_logical_disk_percent_time	Percent time the disk was in read + write operations	Gauge	volume	Custom: Detailed metrics
windows_logical_disk_percent_time	Percent time the disk was idle.	Gauge	volume	Custom: Detailed metrics
windows_logical_disk_percent_free_space	Percent space free on volume.	Gauge	volume	Custom: Detailed metrics
windows_logical_disk_average_disk_sec_per_transfer	Measures the average time of data reads and writes on the disk.	Gauge	volume	Custom: Detailed metrics
windows_logical_disk_average_disk_sec_per_read	Measures the average rate of disk read requests that are executed per second on a specific physical disk.	Gauge	volume	Custom: Detailed metrics
windows_logical_disk_average_disk_sec_per_write	Indicates how fast data is being written on average for a specific logical disk.	Gauge	volume	Custom: Detailed metrics
windows_logical_disk_split_ios_per_sec	Rate the I/Os to the disk were split into multiple I/Os per sec.	Gauge	volume	Custom: Detailed metrics
windows_logical_disk_bytes_per_sec	Exposes the rate bytes are	Gauge	volume	Custom: Detailed metrics

NAME	DESCRIPTION	TYPE	DIMENSIONS	MODE
	transferred to or from the disk during write or read operations per sec.			
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	Custom: Detailed metrics
windows_logical_disk_reads_per_sec	Exposes the rate of read operations on the disk per sec.	Gauge	volume	Custom: Detailed metrics
windows_logical_disk_transfers_per_sec	How fast data is being read and written for a specific logical disk per sec.	Gauge	volume	Custom: Detailed metrics
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	Custom: Detailed metrics

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.

15. Techniques & Tips

15.1. 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 TCP 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:

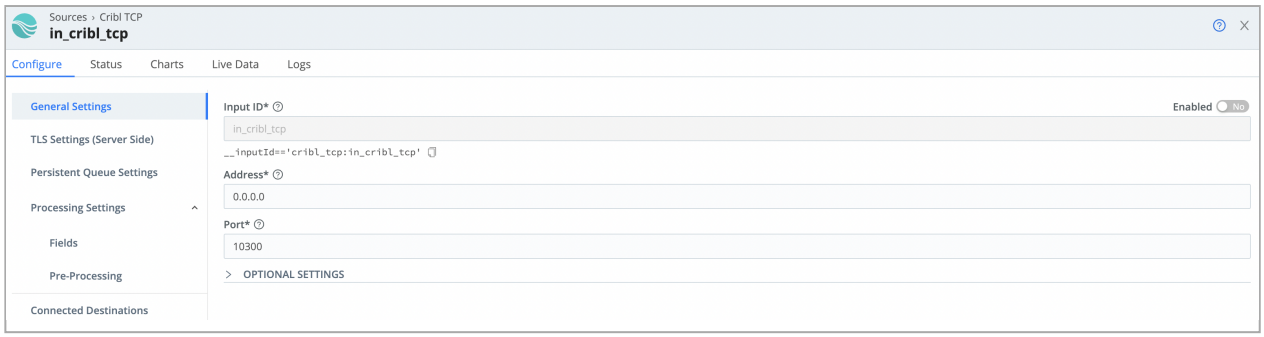
- Configure Cribl TCP Source on Cribl Stream to receive data from the Edge Node.
- Configure the Exec Source on Cribl Edge to collect data on the Edge Node.
- Configure the Cribl TCP Destination on Cribl Edge to send data to Cribl Stream.
- Configure a Route to Send the Data.

And finally, we will confirm the data flow.

> While this use case connects Edge Nodes to Workers through the Cribl TCP Source and Destination, you can also use the Cribl HTTP Source and Destination in certain circumstances – such as when a firewall or proxy blocks raw TCP egress.

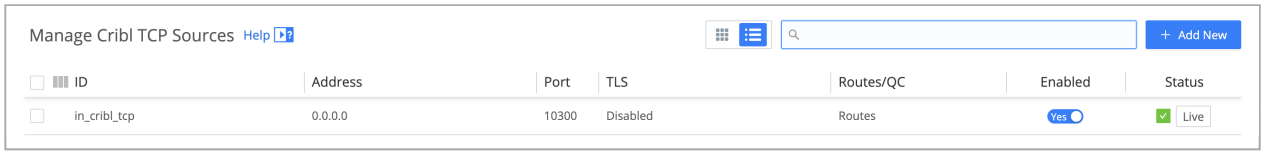
Configure the Cribl TCP Source on Cribl Stream

In Cribl Stream, start by configuring and enabling a Cribl TCP Source. The key requirement here is to set the Port to listen on. By default, the Cribl TCP Destinations listen on Port `10300`. To simplify our scenario, we will set the Cribl TCP Source to listen on the same Port. (Optionally, you can also configure TLS, Event Breakers, metadata fields, and/or a pre-processing Pipeline.)



Configuring a Cribl TCP 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 TCP Source

> On Cribl-managed Cribl.Cloud Worker/Edge Nodes, make sure that TLS is either disabled on both the Cribl TCP Source and the Cribl TCP 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

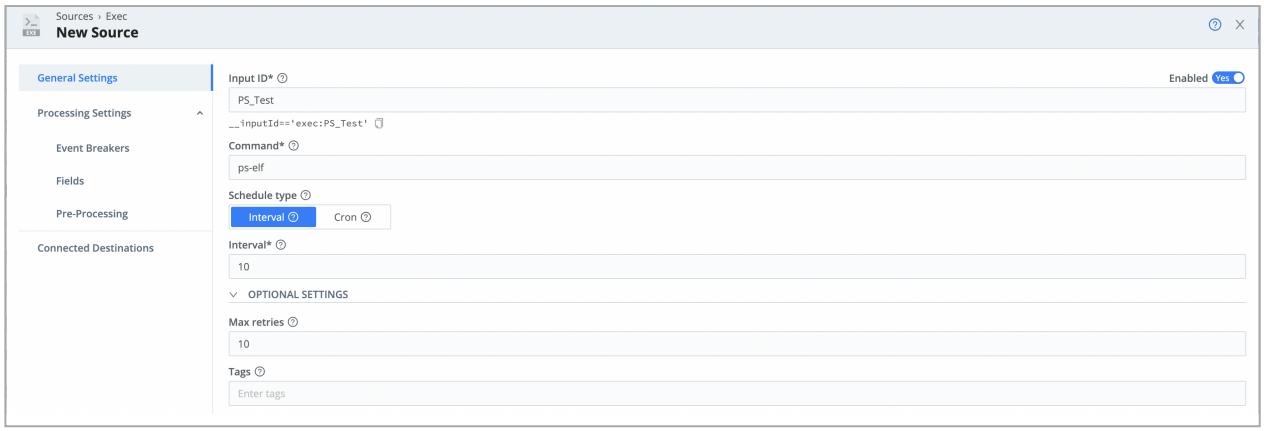
Next, we'll configure the Exec Source on your Edge Node. This Source will break the incoming streams of data into discrete events, and send them to Cribl Stream.

> 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.

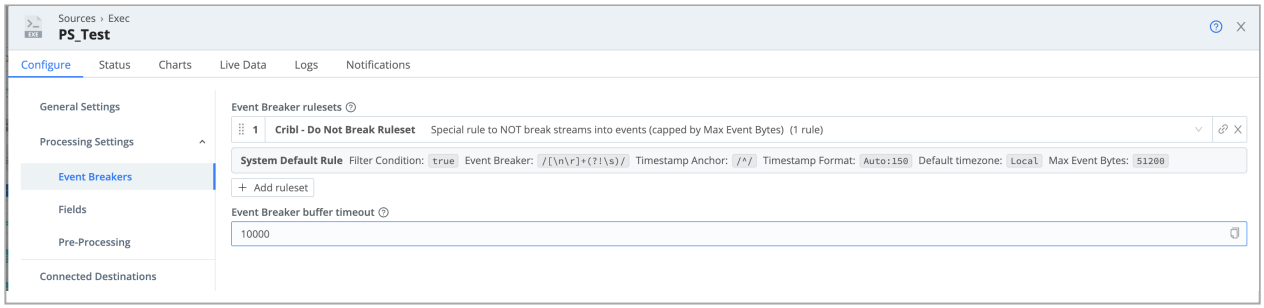
In our example, we are running the `ps` command to list and retrieve running processes every `10` seconds.



Configuring an Exec Source

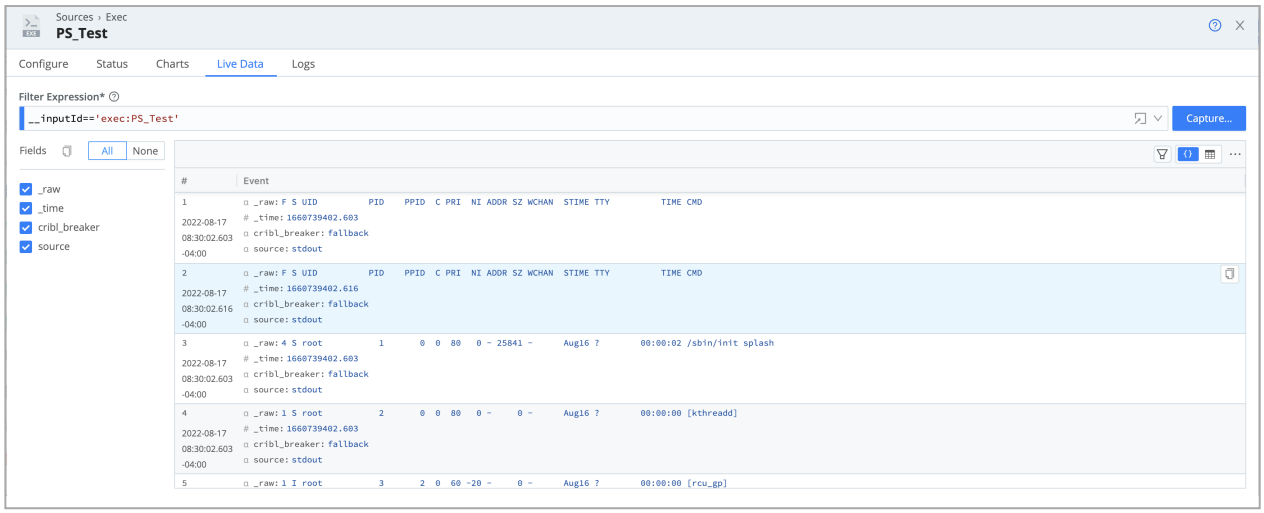
If we don't configure an Event Breaker, then with each capture we run on the dataset, each process will be ingested as its own event, without the header information. So to structure the data, we'll add an Event Breaker.

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.



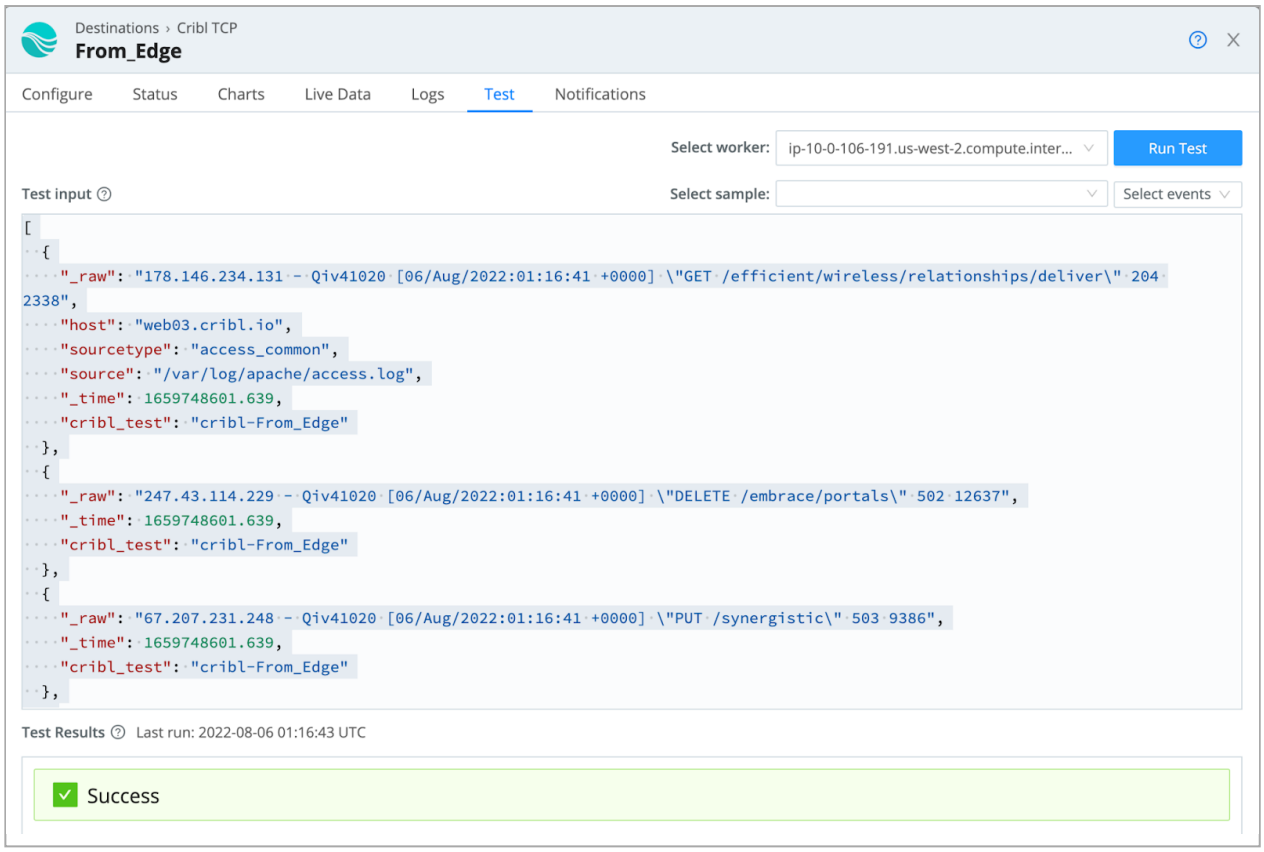
Preview Live Data

Configure the Cribl TCP Destination on Cribl Edge

To get the data flowing, we'll configure the Cribl TCP Destination on your Edge Node. A few things to note when configuring this Destination:

- Set the Port to listen on. For this example, we'll use the default `10300`. If you configure a different Port, make sure the Source points to the same Address and Port.
- If you don't have a load balancer in front of your Workers, you can configure load balancing directly on this Destination.
- Optionally, define your **Compression**, **Throttling**, and **Backpressure behavior** requirements.
- On Cribl-managed Cribl.Cloud Worker/Edge Nodes, 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.

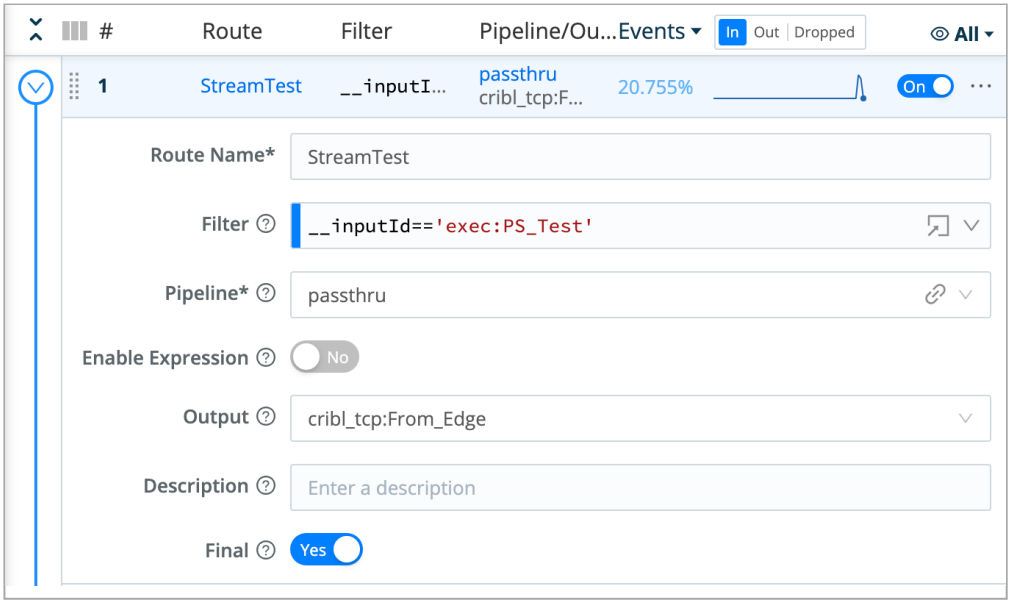
Once you've configured your Destination, test it to verify that your Edge Node can communicate with the Stream Worker Group.



Testing your Destination

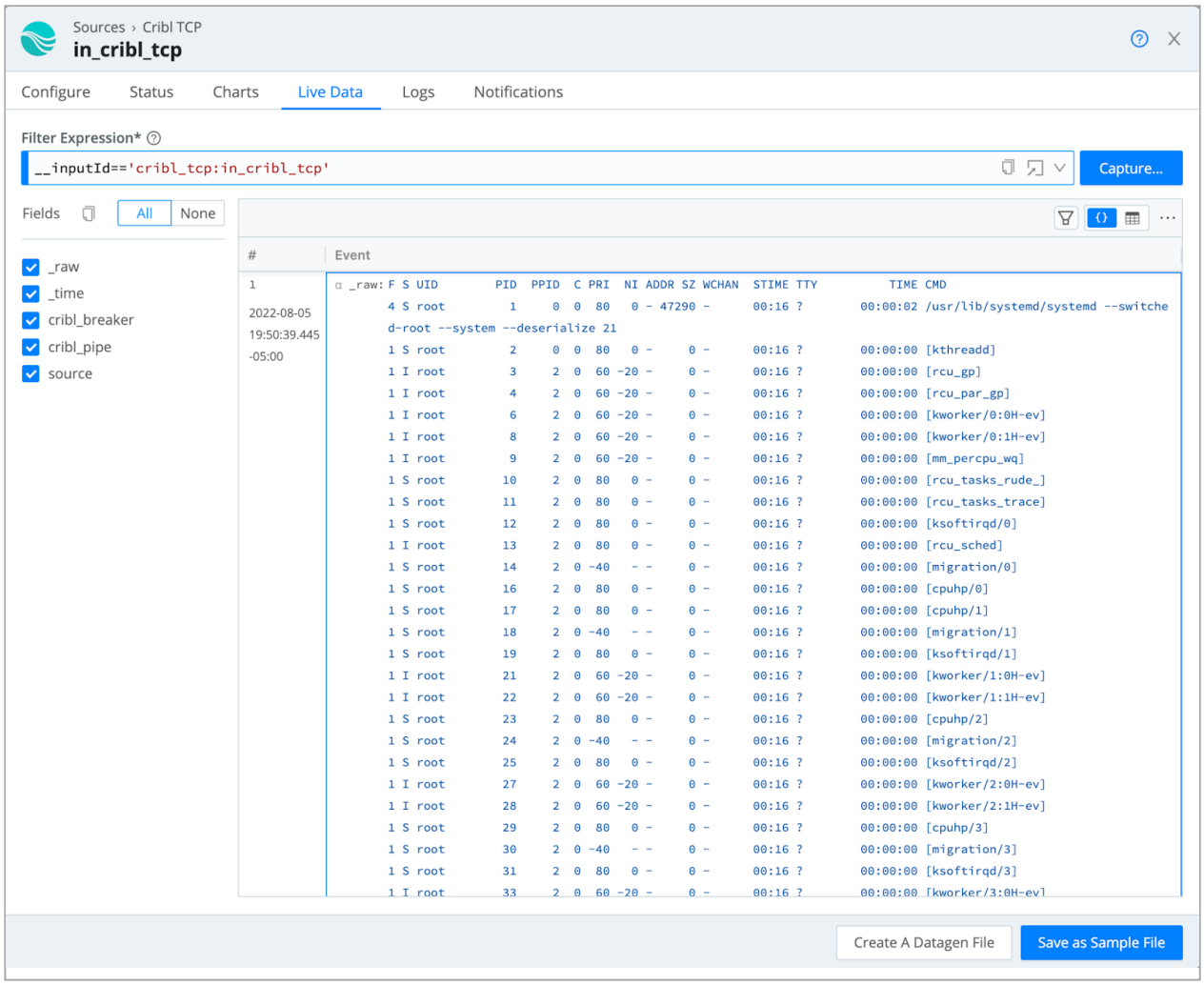
Configure a Route to Send the Data

Finally, configure a Route to send your data to Cribl Stream. In this example, we are using the `passthru` Pipeline.



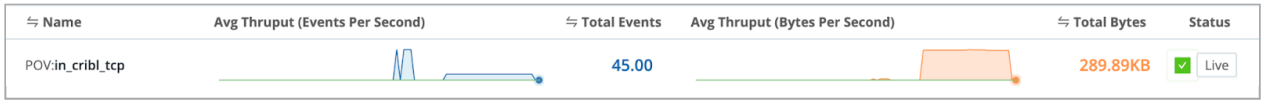
Confirm the Data Flow

To confirm that your data is flowing, navigate back to Cribl Stream's Cribl TCP Source. Run a **Live Data** capture on the Source.



Data flow in the Source

You can also check the Monitoring page's **Data** submenu, to isolate the throughput on your Source.



Monitoring the Source throughput

15.2. Cribl Edge as a Sidecar Container in AWS ECS

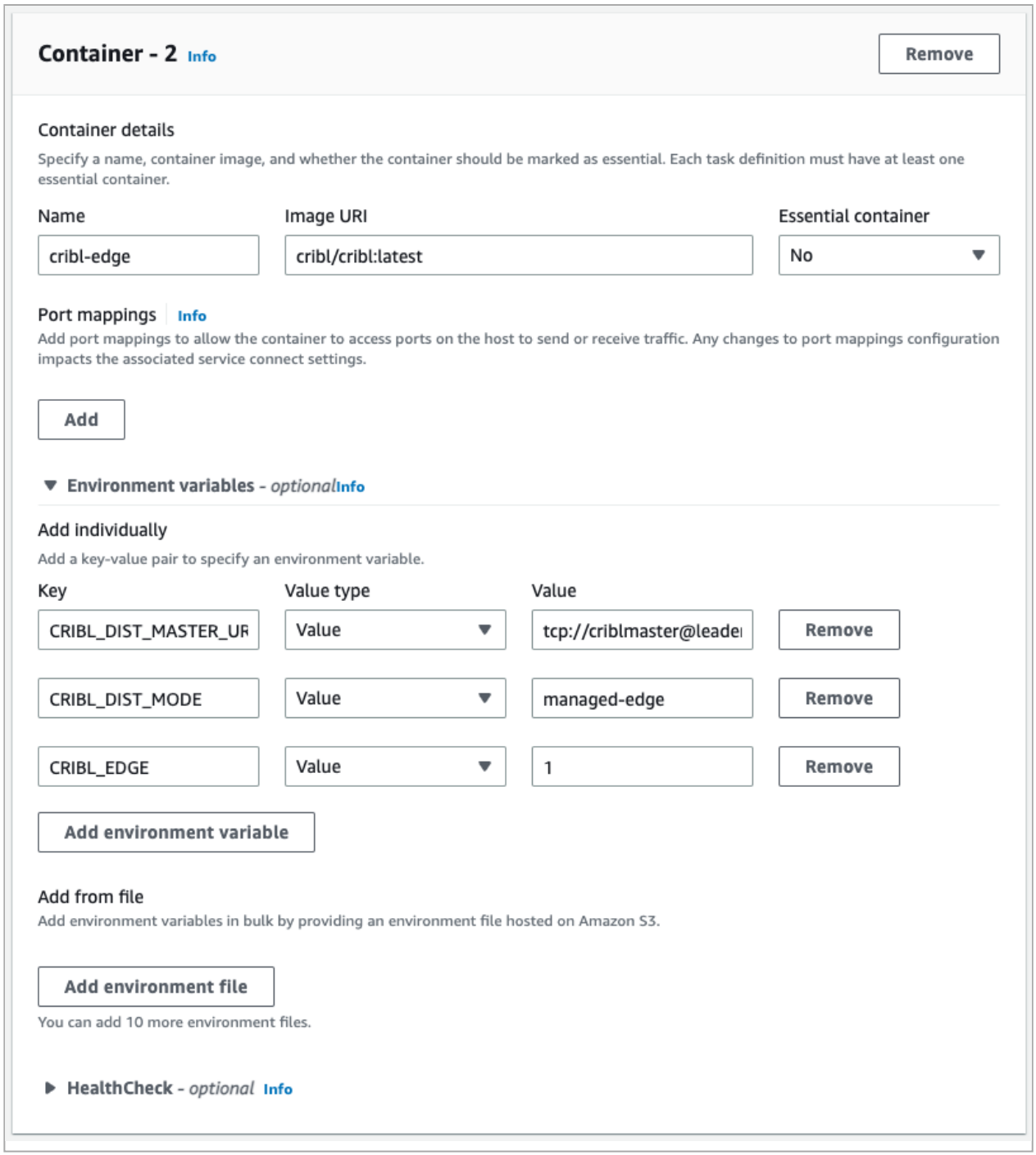
You can use Cribl Edge to collect logs from a service or task running in ECS. This is an alternative to forwarding logs to CloudWatch Logs.

> 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

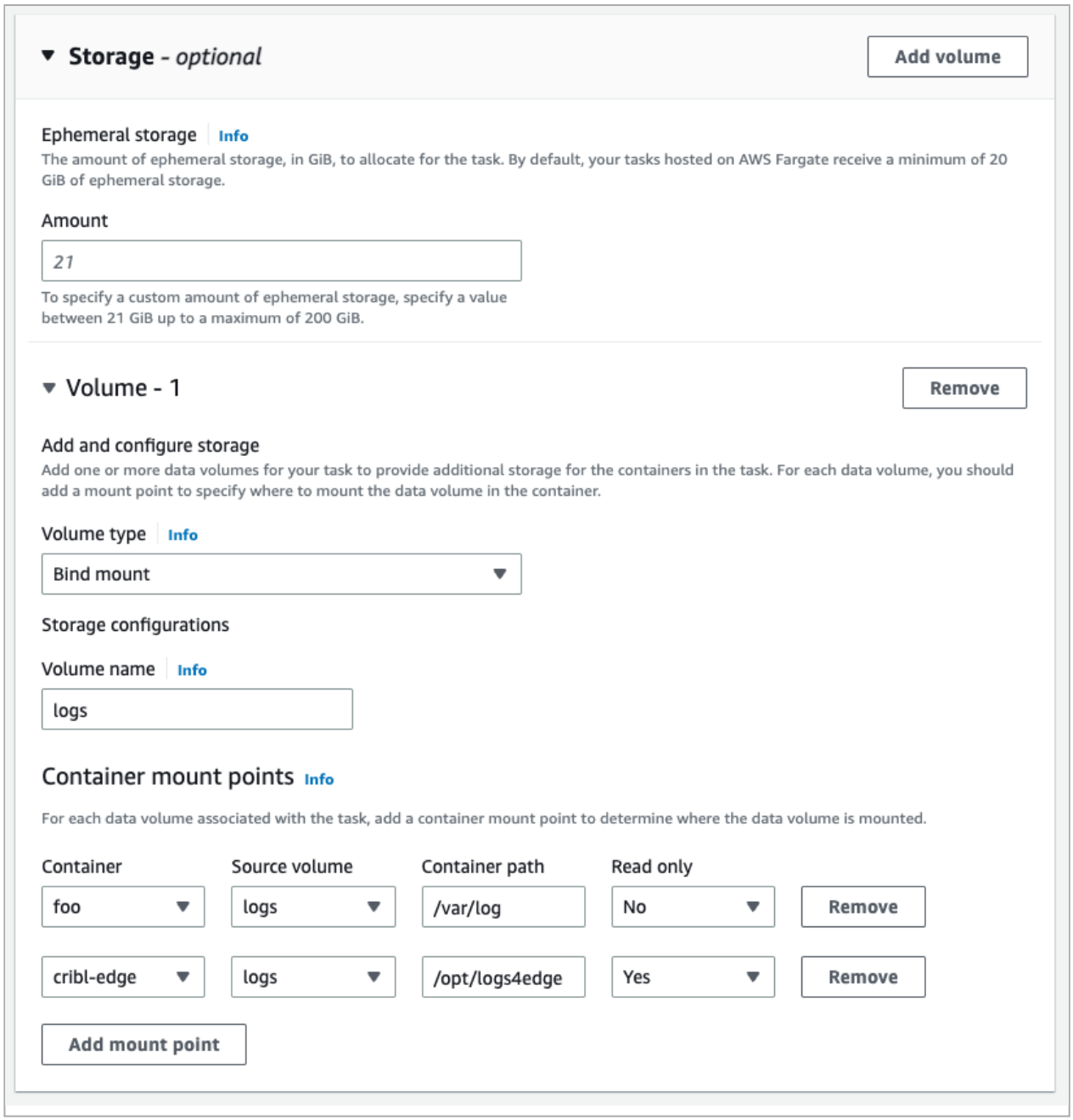


Add Environment Variables to Container Details

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.

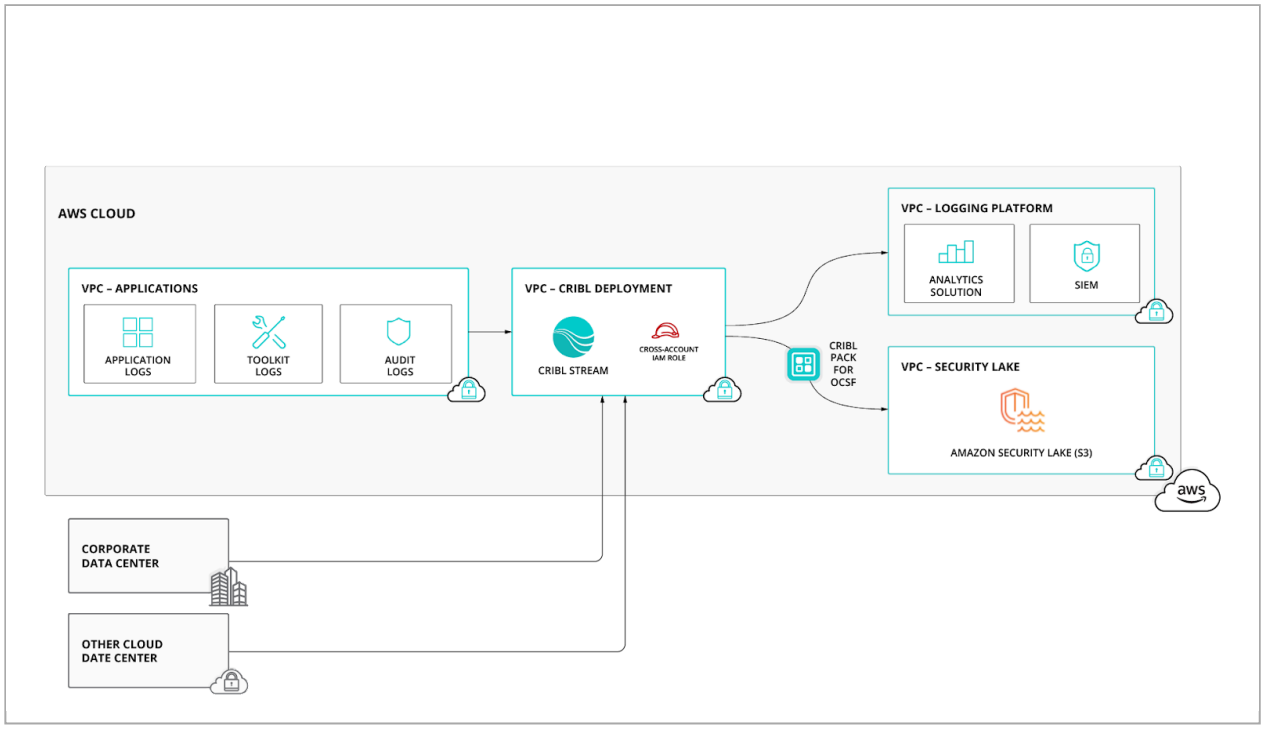
```
{
    "containerDefinitions": [
        {
            "name": "whatever",
            "image": "foo/bar:latest",
            ...
            "mountPoints": [
                {
                    "sourceVolume": "logs4edge",
                    "containerPath": "/var/log",
                    "readOnly": 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 ...
}
```

15.3. 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.

Setting 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.

Setting 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 to open the Portal Page.
2. In the top nav, click **Network Settings** and open the **Trust** tab.
3. Click 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. In the top nav, click **Cribl.Cloud** to return to the Portal Page.
2. Click **Manage Stream** or **Manage Edge** to open the **Fleets** page.
3. Click 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.) This takes you to the **Manage** > **Overview** page.
4. Navigate to **Processing** > **Packs** and click **Add Pack**. From the drop-down, select **Import from Git** to open the **Import** modal.
5. For the **URL**, enter:

```
https://github.com/asc-me-cribl/cribl_ocsf_postprocessing
```

6. For the **New Pack ID**, enter:

```
Cribl-OCSF_Post_Processor
```

7. Click **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.

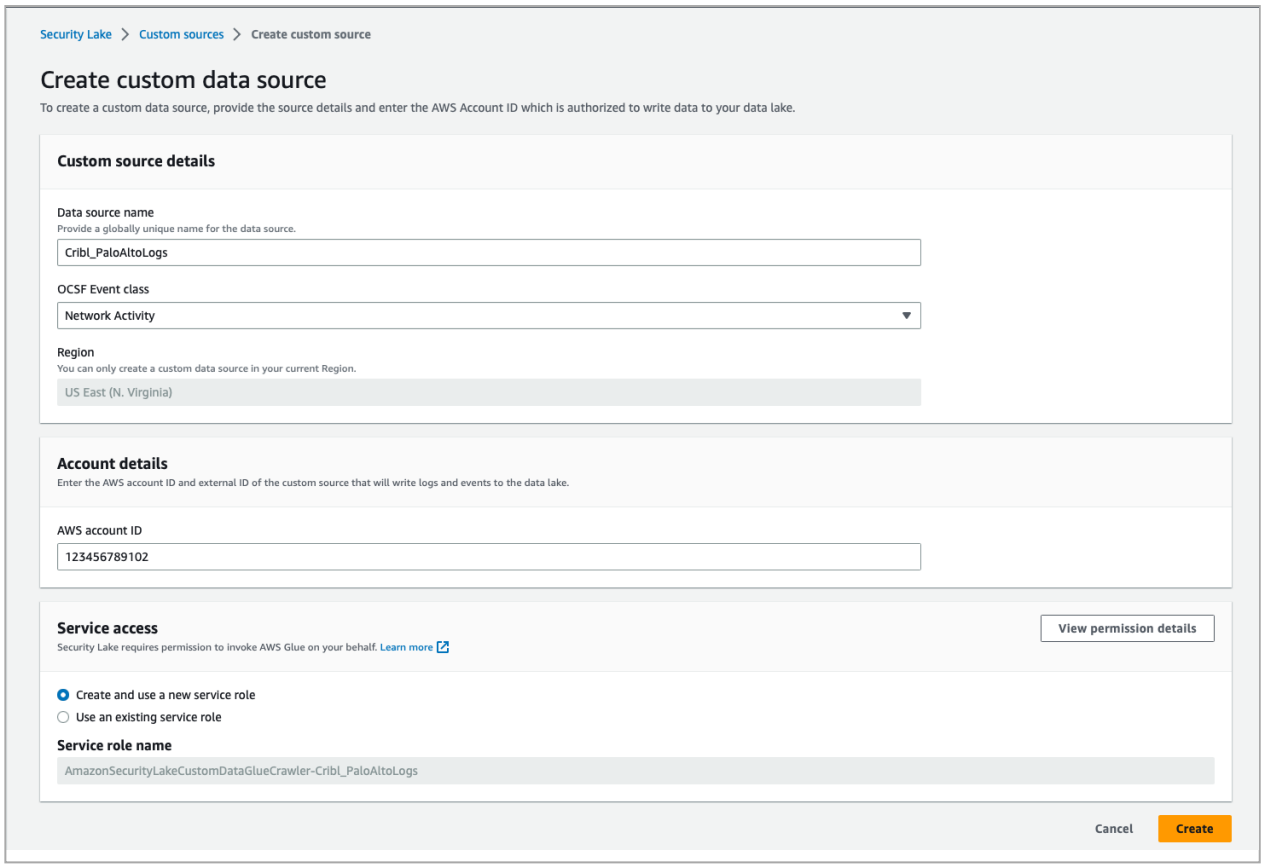
Creating an Amazon Security Lake Custom Source

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. Click 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, click 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.

Creating 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.

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 click **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 – i.e., **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.

Connecting 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 clicking the Syslog Source tile, click **Select Existing**, then click the pre-configured `in_syslog` Source. When prompted to `Switch in_syslog to send to QuickConnect instead of Routes?`, click `Yes`.

Click **+** 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.

Sending 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 in.main-default-<cribl_org_id>.cribl.cloud 10070
```

Going 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`

DATA SOURCE	OCSF EVENT CLASSES HANDLED	CRIBL SOURCE	CRIBL PACK DISPLAY NAME	PARQUET SCHEMA(S)
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`
				`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`

15.4. SSO with Okta and OIDC

Cribl Edge supports SSO/OIDC user authentication (login/password) and authorization (user's group membership, which you can map to Cribl Roles). Using OIDC will change the default `Log in` button on the login page to a button labeled `Log in with OIDC` which redirects to the configured Identity Provider (IDP).

If you are a Cribl Edge admin and want to offer single sign-on (SSO) to your users, you can choose OpenID Connect (OIDC) as the authentication type, then configure an IDP to use OIDC within its single sign-on flow. Once configuration is complete (several steps later), the Cribl Edge login page will send users to the IDP's login UI. Besides the IDP, some settings will refer to the Service Provider (SP), which in this context is your Cribl Edge instance.

The IDP can be Okta or Google, among others. Configuring SSO requires going back and forth between Cribl Edge and the IDP's UI. In this page, we walk through the process for configuring SSO with OIDC, using Okta as the IDP.

The last part of the walkthrough is a complete description of Cribl Edge's **Authentication** settings (with OIDC selected as the authentication method), they're here. Skip directly to this section if you just need a UI reference, or if your IDP is not Okta. For non-Okta deployments, please join us on Cribl's Community Slack at https://cribl-community.slack.com/ and share your questions.

> Make sure you don't get locked out of Cribl Edge! Enable the **Allow local auth** toggle until you're certain that external auth is working as intended. If you do get locked out, refer to Manual Password Replacement for the remedy.
>
> Like other external auth methods, OpenID Connect requires either an Enterprise or a Standard license. It is not supported with a Free license.

Plan Your Mapping of Okta Groups to Cribl Edge Roles

In Okta, admins organize their users in groups. In Cribl Edge, there are no user groups, but there are Roles. Your task includes **mapping** 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 applied.
- 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.

The example below illustrates how multiple mappings work: The groups in Mapping **b** and **c** each map to multiple Roles, while both the `reader_all` and `editor_cloud` Roles map to multiple groups.

MAPPING	OKTA GROUP	CRIBL EDGE ROLE(S)
a.	Cribl Admins	`admin`
b.	Cloud Admins	`reader_all`, `editor_cloud`
c.	Security Team	`reader_all`, `editor_cloud`, `editor_firewall`

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.

Integrate Okta with Cribl Edge

1. Log in to your Okta tenant admin console.
2. In the left nav, select **Applications** > **Applications**.
3. Click **Create App Integration**.
    - For **Sign-in method**, select `OIDC - OpenID Connect`.
    - For **Application type**, select `Web Application`.
4. Click **Next** to open the **New Web App Integration** page.
    - In the **App name** field, enter Cribl Edge.
    - (Optional) In the **Logo** field, upload the Cribl logo. You can use a logo from the Cribl Media Kit.
    - (Optional) If you wish to keep your OIDC app hidden, check the **App visibility** check box.

5. In the **Sign-in redirect URIs** field, replace the default with your Leader base URL, and with `/api/v1/auth/authorization-code/callback` as the path. This is the Cribl Edge callback API endpoint.

6. (Optional) In the **Sign-out redirect URIs** field, append `/login` to the pre-filled path.

7. In the **Assignments** > **Controlled access** area:

    - If all your Okta users need access to Cribl Edge, select **Allow everyone in your organization to access**.
    - To permit specific Okta groups to access Cribl Edge, 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.

8. Click **Save**.

Okta should show an `Application Created Successfully` message.

New Web App Integration

General Settings

App integration name

Cribl Stream

Logo (Optional)

Grant type

Learn More

Client acting on behalf of itself

- [ ] Client Credentials

Client acting on behalf of a user

- [x] Authorization Code
- [ ] Refresh Token
- [ ] Implicit (Hybrid)

Sign-in redirect URIs

Okta sends the authentication response and ID token for the user's sign-in request to these URIs

Learn More

https://leader:9000/api/v1/auth/authorization-code/callback

+ Add URI

Sign-out redirect URIs (Optional)

After your application contacts Okta to close the user session, Okta redirects the user to one of these URIs.

Learn More

https://leader:9000/login

+ Add URI

Trusted Origins

Base URIs (Optional)

Required if you plan to self-host the Okta Sign-In Widget. With a Trusted Origin set, the Sign-In Widget can make calls to the authentication API from this domain.

Learn More

+ Add URI

Assignments

Controlled access

- ( ) Allow everyone in your organization to access
- (•) Limit access to selected groups

Cribl Admins ×

Save Cancel

Copy Your Okta App's Client ID and Client Secret

In the **Client Credentials** panel, copy both the **Client ID** and **Client Secret**, and temporarily store them locally. You will need them in the next step, when you configure Cribl Edge.

Begin Configuring OIDC Auth in Cribl Edge

> The only OAuth 2.0 flow that Cribl Edge supports is the Authorization Code Grant flow.
>
> In version 3.0 and higher, Cribl Edge's former "master" application components are renamed "leader." Above, while some legacy terminology remains within URLs, this document will reflect that.

In Cribl Edge, select **Settings** > **Access Management** > **Authentication**.

1. Choose `OpenID Connect` from the **Type** dropdown.
2. Choose `Okta` from the **Provider** dropdown.
3. In the **Audience** field, enter your Cribl Edge UI base URL. Do **not** append a trailing slash.
4. In the **Client ID** and **Client secret** fields, enter the respective values that you copied from the Okta UI in the previous step.
5. 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, which is: `openid profile email`.
6. Obtain the authentication, token, user info, and logout URLs for your Okta app, by sending a request to the OpenID Connect Discovery endpoint.

   - This endpoint has the URL:
     `https://<tenant>.okta.com/.well-known/openid-configuration`

     ...where `<tenant>` is your Okta tenant name. For example:
     `https://dev-12345678.okta.com/.well-known/openid-configuration`

- You can view the discovery document in your web browser, or use jq to extract the needed values, as in the following example:

  ```
  curl -s https://<tenant>.okta.com/.well-known/openid-configuration | jq '. | {"auth": (.authorization_endpoint), "token":(.token_endpoint), "userinfo": (.userinfo_endpoint), "logout": (.end_session_endpoint)}'
  ```

- Sample response:

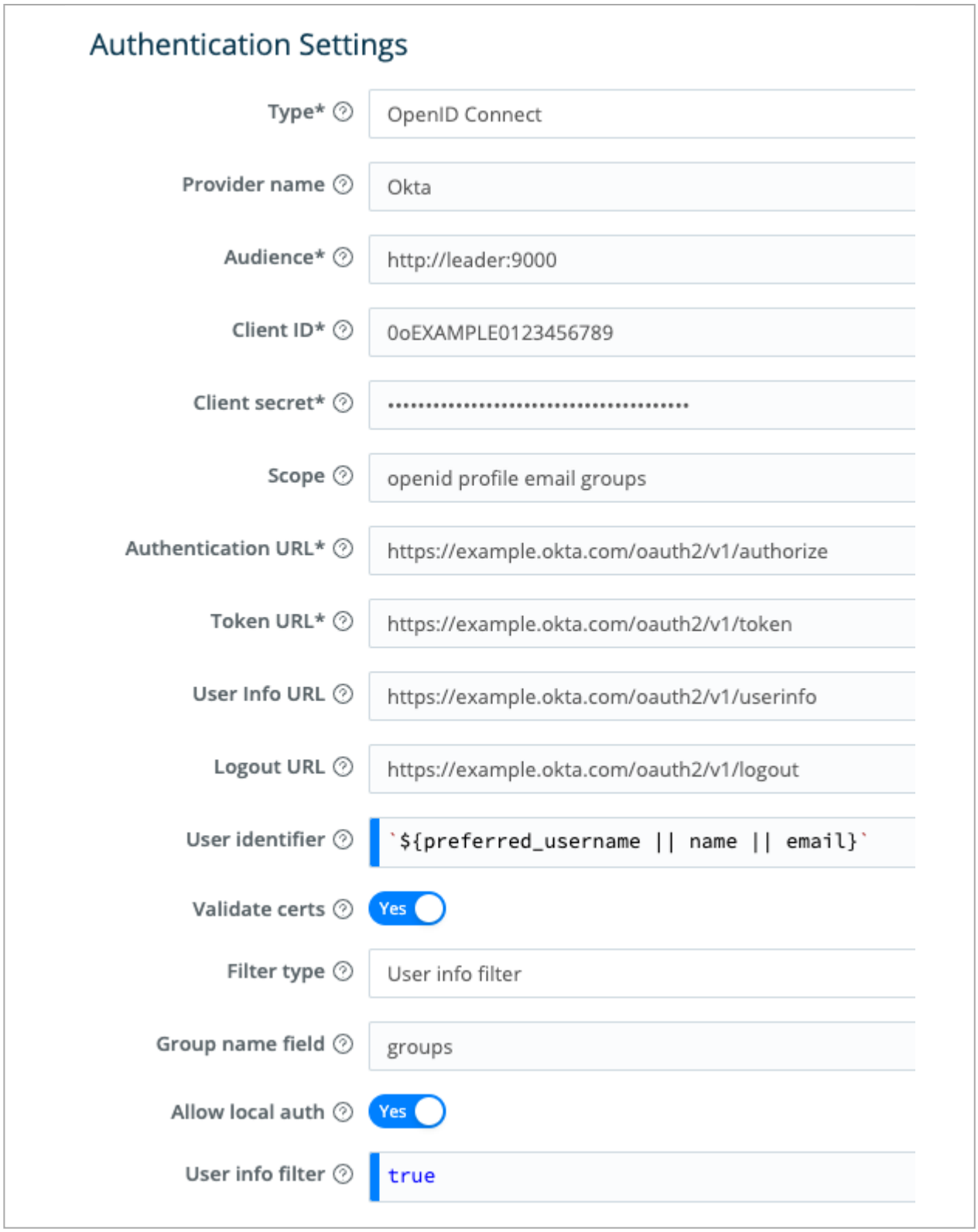
  ```
  { "auth": "https://dev-416897.oktapreview.com/oauth2/v1/authorize", "token": "https://dev-416897.oktapreview.com/oauth2/v1/token", "userinfo": "https://dev-416897.oktapreview.com/oauth2/v1/userinfo", "logout": "https://dev-416897.oktapreview.com/oauth2/v1/logout" }
  ```

7. Populate the **Authentication URL Token URL** fields with the respective `auth` and `token` URLs.

8. If you configured Okta to use groups, populate the **User info URL** field with the `userinfo` URL.

   This is necessary because Okta does not send group information in the `id_token` passed to Cribl Edge.

9. If you want **Account** > **Log out** in Cribl Edge to log the user out **globally**, populate the **Logout URL** field with the `logout` URL. This means that when a user clicks the **Accounts** > **Log out** link in Cribl Edge, they are logged out of **both** Cribl Edge and Okta.



Authentication settings in Cribl Edge

Configure Response to Okta `/userinfo` Endpoint

An Okta tenant's user groups can be mastered either inside Okta, outside Okta, or both.

When the `/userinfo` endpoint is queried, Okta returns the appropriate groups membership of the user back to Cribl Edge:

- For groups mastered inside Okta only, the app should pass a `Filter` type groups claim to Cribl Edge.
- For groups mastered outside Okta (e.g., Active Directory), or both inside and outside, the app should pass an `Expression` type groups claim back to Cribl Edge.

See the Okta documentation on dynamic allow lists and using Okta together with Active Directory.

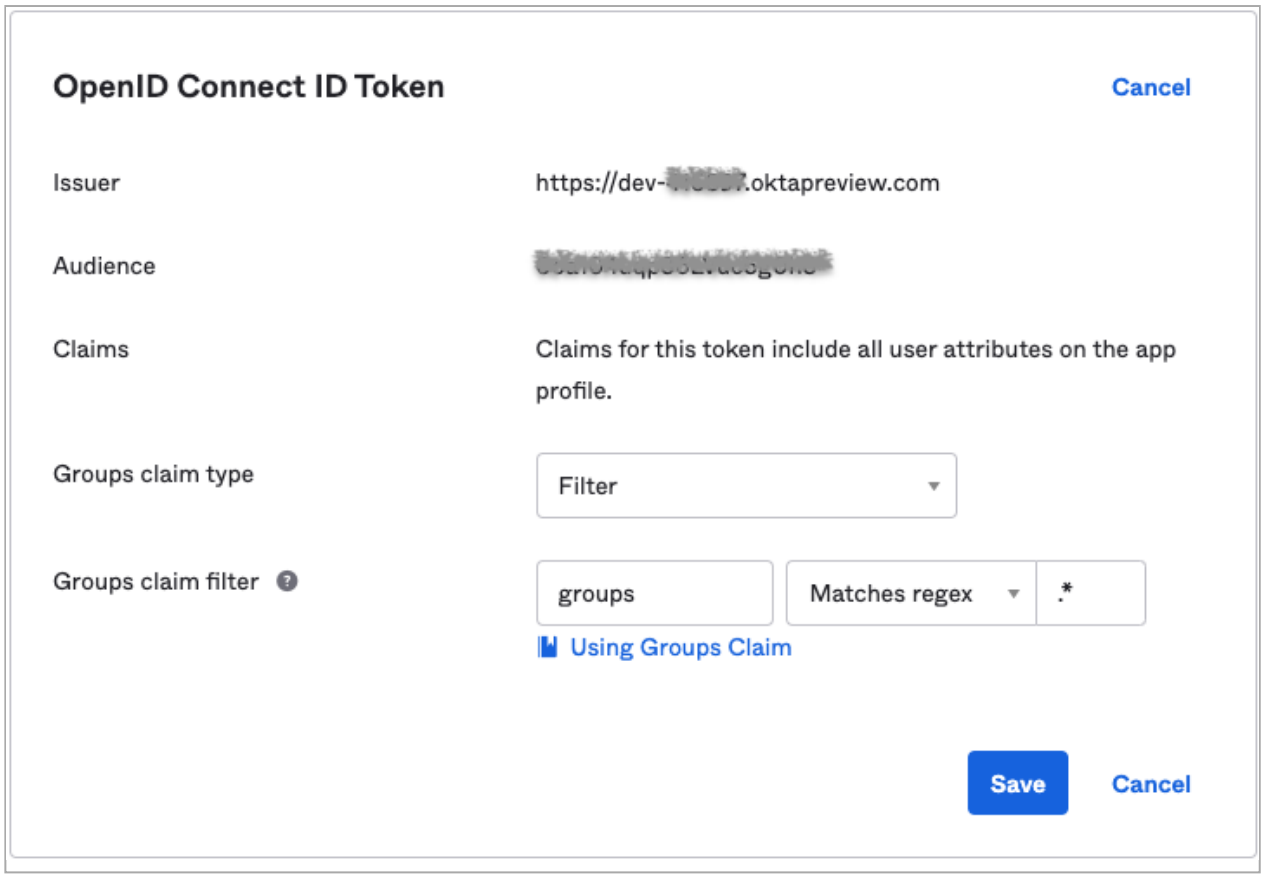
In Okta, you should still be in the panel for the app you created. If not, you can get there by opening **Applications** > **Applications** and selecting the app.

- For groups mastered **inside** Okta only, complete this procedure.
- For groups mastered **outside** Okta, or both inside and outside, complete this procedure.

Configure Groups Inside of Okta

Open the **Sign On** tab. Then, in the **OpenID Connect ID Token** panel:

1. Click **Edit** to change the value of **Groups claim filter** to `groups` and show filter options.
2. Leave **Groups claim type** set to `Filter`.
3. Choose **Matches regex** from the dropdown, and enter `.*` as the regex.
4. Click **Save**.



Role mapping, beginning

Configure Groups Outside of Okta

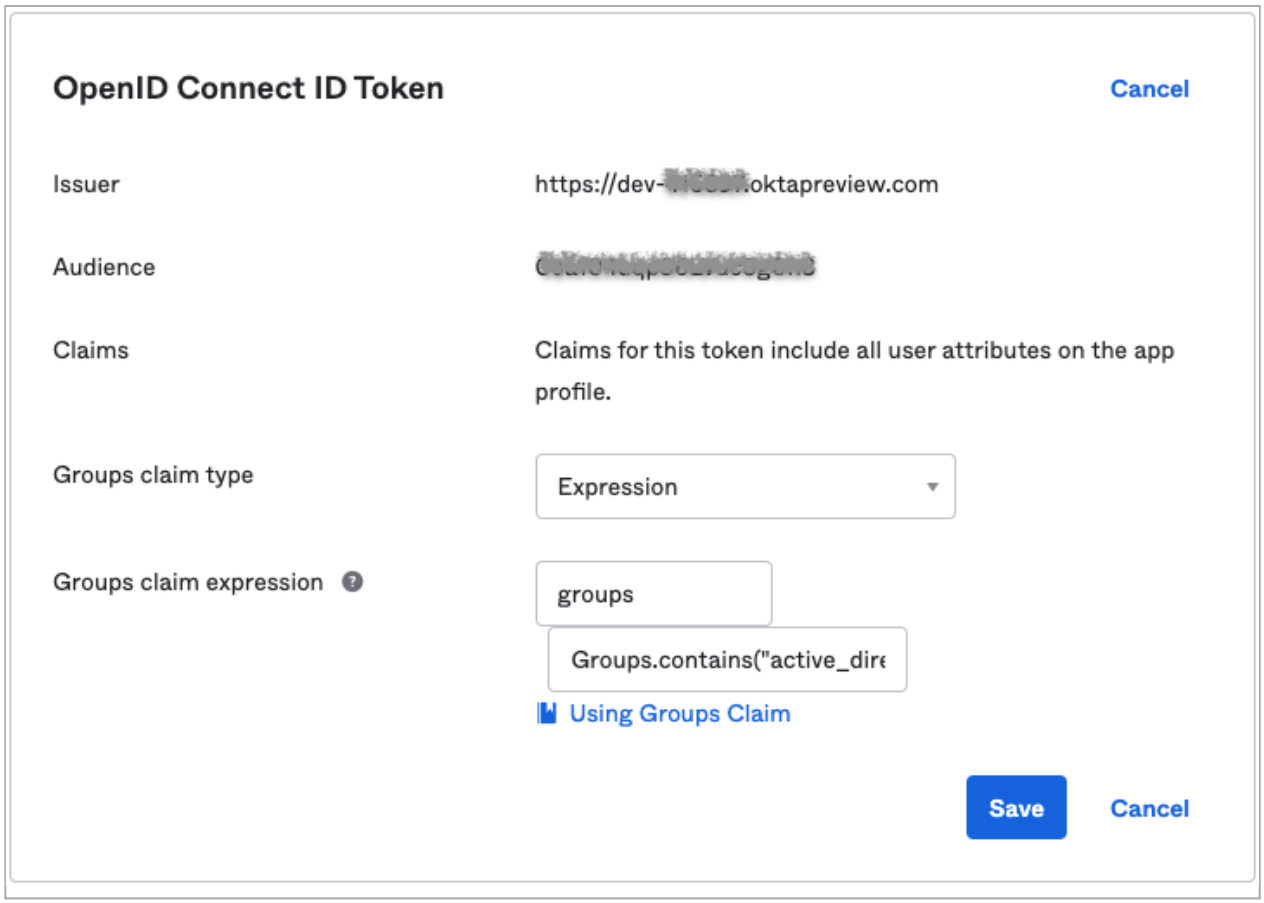
Open the **Sign On** tab, if necessary. Then, in the **OpenID Connect ID Token** panel:

1. Click **Edit** to change the value of **Groups claim filter** to `groups` and show filter options.
2. Set **Groups claim type** set to `Expression`.
3. In the **Groups claim expression**, enter an expression field that matches the groups you want passed to Cribl Edge. See the Okta documentation for more details.

   For example, to match on Active Directory groups that contain the string `okta`, use the following expression:

   ```
   Groups.contains("active_directory", "cribl", 10)
   ```

4. Click **Save**.

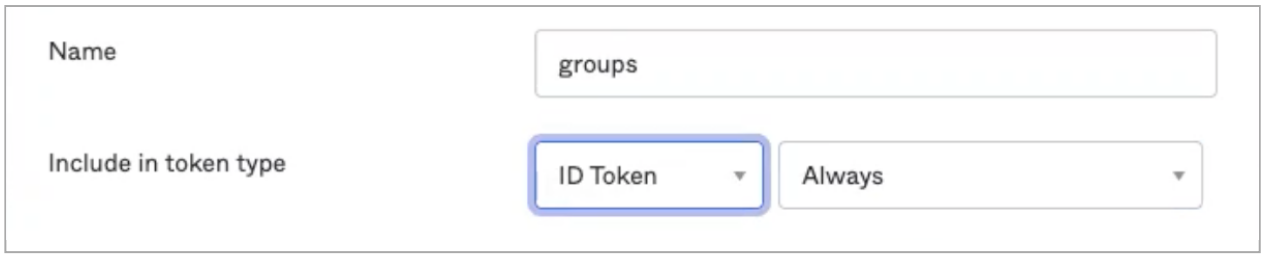


Role mapping, continued

Configure ID Token to Include Groups Claim

> Okta can recognize your groups **only** if your ID token includes your groups claim, as you'll configure here.

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, click 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).

Finish Configuring OIDC Auth in Cribl Edge

> This section documents the entire Cribl Edge Authentication UI. If you have been working through the procedures from earlier in this page, you will have completed some of the following steps already.

Navigate to **Settings** > [**Global Settings** >] **Access Management > Authentication > Type** and select **OpenID Connect**. This exposes the following controls.

- **Provider name**: The name of the identity provider service. You can select **Google** or **Okta**, both supported natively. Manual entries are also allowed.
- **Audience (SP entity ID)**: The base URL, e.g.: `https://leader.yourDomain.com:9000` for a distributed environment. For the IDP, this serves as a unique identifier for the SP (i.e., your Cribl Edge instance).

  > For distributed environments with a second Leader configured, modify the **Audience** field to point to the load balancer instead of the Leader Node.

- **Client ID**: The `client_id` from provider configuration.
- **Client secret**: The `client_secret` from provider configuration.
- **Scope**: Space-separated list of authentication scopes. The default list is: `openid profile email`. If you populate the **User info URL** field, you must add `groups` to this list.
- **Authentication URL**: The full path to the provider's authentication endpoint. Be sure to configure the callback URL at the provider as `<masterServerFQDN>:9000/api/v1/auth/authorization-`

`code/callback`, e.g.: `https://leader.yourDomain.com:9000/api/v1/auth/authorization-code/callback`.

- **Token URL**: The full path to the provider's access token URL.
- **User info URL**: The full path to the provider's user info URL. Optional; if not provided, Cribl Edge will attempt to gather user info from the ID token returned from the **Token URL**.
- **Logout URL**: The full path to the provider's logout URL. Leave blank if the provider does not support logout or token revocation.
- **User identifier**: JavaScript expression used to derive `userId` from the `id_token` returned by the OpenID provider.
- **Validate certs**: Whether to validate certificates. Defaults to `Yes`. Toggle to `No` to allow insecure self-signed certificates.
- **Filter type**: Select either **Email allowlist** or **User info filter**. This selection displays one of the following fields:
  - **Email allowlist**: Wildcard list of emails/email patterns that are allowed access.
  - **User info filter**: JavaScript expression to filter against user profile attributes. E.g.: `name.startsWith("someUser") && email.endsWith("domain.com")`
- **Group name field**: Field in the **User info URL** response (if configured); otherwise, `id_token` that contains the user groups. Defaults to `groups`.
- **Allow local auth**: Toggle to `Yes` to also users to log in using Cribl Edge's local authentication. This enables an extra button called `Log in with local user` on the Cribl Edge login page. (This option ensures fallback access for local users if SSO/OpenID authentication fails.)

> 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.

- **Email allowlist**: Wildcard list of emails/email patterns that are allowed access.

Note the following details when filling in the form – for example, when using Okta:

- `<Issuer URI>` is the account at the identity provider.
- `Audience` is the URL of the host that will be connecting to the Issuer (e.g., `https://leader.yourDomain.com:9000` for a distributed environment). The issuer (Okta, in this

example) will redirect back to this site upon authentication success or failure.

- `User info URL` is required, because Okta doesn't encode groups in `id_token`. Azure AD and Google also rely on this field.

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**: Add a mapping for each external user group that you want to map to one or more Cribl roles. Then enter the group and select the role(s).

See also the explanation of role mapping above.

Role Mapping Example

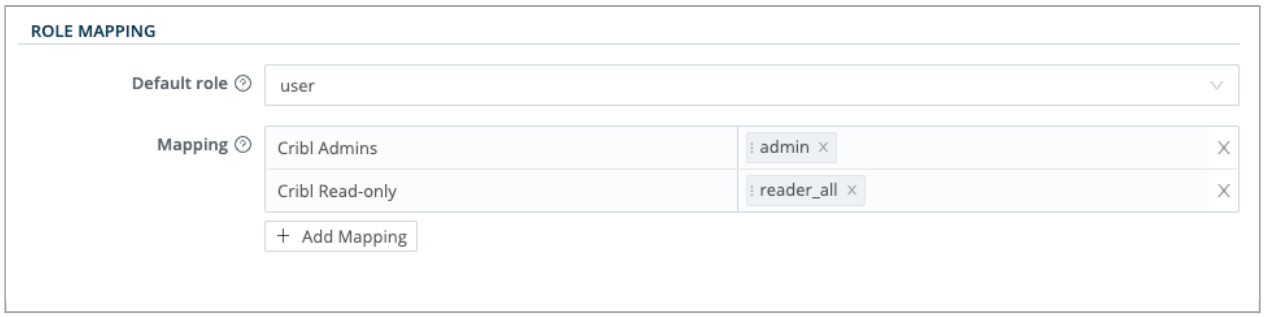
Role mapping UIs will differ from one IDP to another. For this example, we'll look at Okta's.

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, select **Settings** > **Access Management** > **Authentication**.
2. Scroll down to the **ROLE MAPPING** section.

   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).



Role mapping, concluded

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. Click **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.

Troubleshooting Resources

Cribl University's Troubleshooting Criblet on SSO Integration for Stream On-Prem - Okta walks you through this whole configuration flow. To follow the direct course link, first log into your Cribl University account. (To create an account, click 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.

15.5. SSO with Okta and SAML

Cribl Edge supports SSO/SAML user authentication (login/password) and authorization (user's group membership, which you can map to Cribl Roles). Using SAML will change the default `Log in` button on the login page to a button labeled `Log in with SAML 2.0` which redirects to the configured Identity Provider (IDP).

If you are a Cribl Edge admin and want to offer single sign-on (SSO) to your users, you can choose SAML 2.0 as the authentication type, then configure an IDP to use SAML within its single sign-on flow. Once configuration is complete (several steps later), the Cribl Edge login page will send users to the IDP's login UI. Besides the IDP, some settings will refer to the Service Provider (SP), which in this context is your Cribl Edge instance.

The IDP can be Okta or Google, among others. Configuring SSO requires going back and forth between Cribl Edge and the IDP's UI. In this page, we walk through the process for configuring SSO with SAML, using Okta as the IDP.

The last part of the walkthrough is a complete description of Cribl Edge's **Authentication** settings (with SAML 2.0 selected as the authentication method) is here. Skip directly to this section if you just need a UI reference, or if your IDP is not Okta. For non-Okta deployments, please join us on Cribl's Community Slack at https://cribl-community.slack.com/ and share your questions.

> Make sure you don't get locked out of Cribl Edge! Enable the **Allow login as Local User** toggle until you're certain that external auth is working as intended. If you do get locked out, refer to Manual Password Replacement for the remedy.
>
> SSO with SAML is supported only in Cribl Edge versions 4.1.0 and later.
>
> Like other external auth methods, SAML requires either an Enterprise or a Standard license. It is not supported with a Free license.

Plan Your Mapping of Okta Groups to Cribl Edge Roles

In Okta, admins organize their users in groups. In Cribl Edge, there are no user groups, but there are Roles. Your task includes **mapping** 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 applied.

- 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.

The example below illustrates how multiple mappings work: The groups in Mapping **b** and **c** each map to multiple Roles, while both the `reader_all` and `editor_cloud` Roles map to multiple groups.

MAPPING	OKTA GROUP	CRIBL EDGE ROLE(S)
a	Cribl Admins	`admin`
b	Cloud Admins	`reader_all`, `editor_cloud`
c	Security Team	`reader_all`, `editor_cloud`, `editor_firewall`

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.

Begin Configuring SAML Auth in Cribl Edge

Navigate to **Settings** > [**Global Settings** >] **Access Management > Authentication > Type** and select **SAML 2.0**.

In the **Audience (SP entity ID)** field, enter the base URL of your Cribl Edge instance, e.g., `https://yourDomain.com:9000`. Do **not** append a trailing slash.

> For distributed environments with a second Leader configured, modify the **Audience** field to point to the load balancer instead of the Leader Node.

This will populate three more fields:

- **Sign-on callback URL**

- **Logout callback URL**
- **Metadata URL**

Note the values of all four fields for use in the next section.

If you want the requests that Cribl Edge sends to the IDP to be signed, select or create a **Request certificate**.

- Note the public key for the certificate that you select or create, for use in the next section.

Integrate Okta with Cribl Edge

In this section, we'll create an Okta app that uses SAML and integrates with Cribl Edge.

Configure General Settings (Part 1)

1. Log in to your Okta tenant admin console.
2. In the left nav, select **Applications** > **Applications** to open the Part **1**, **General Settings** page.
3. Click **Create App Integration**.
    - For **Sign-in method**, select `SAML 2.0`.
4. Click **Next** to open the **Create SAML Integration** page.
    - In the **App name** field, enter Cribl Edge.
    - (Optional) In the **Logo** field, upload the Cribl logo. You can use a logo from the Cribl Media Kit.
    - (Optional) If you wish to keep your SAML app hidden, check the **App visibility** check box.

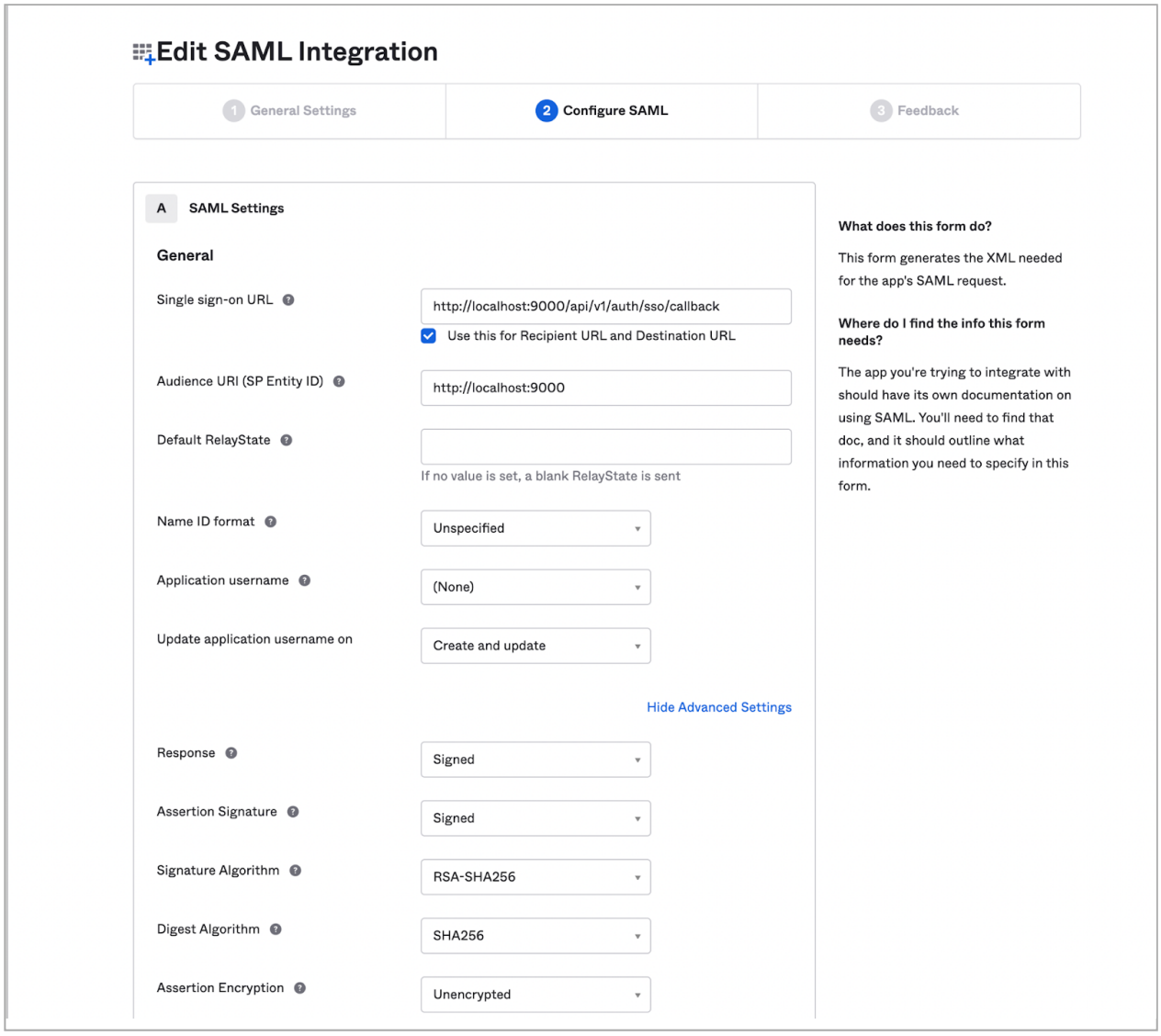
Configure SAML (Part 2)

1. Click **Next** to open the **Configure SAML** > part **A**, **SAML Settings** page. Configure these settings as follows:
    - **Single sign-on URL**: Enter the Cribl Edge **Sign-on callback URL**.
    - **Audience URI (SP Entity ID)**: Enter the Cribl Edge **Audience (SP entity ID)**.
    - **Default RelayState**: Leave blank.
    - **Name ID format**: Leave as `Unspecified` (the default).

- **Application username**: Specify a username; can be a plain username, an email, or a custom username. In the SAML assertion's `subject` statement, this is the value for `NameID`. By default, Cribl Edge will use this value as the username in Stream. Alternatively, you can set a custom attribute statement in Okta (as described in Step 2 below), then set the **Username field** in Cribl Edge to use that instead.

- **Update application username on**: Leave as is (`Create and update`).

- (Optional) Click **Show Advanced Settings** if you want to configure Single Logout, SAML response encryption, etc., as described below.

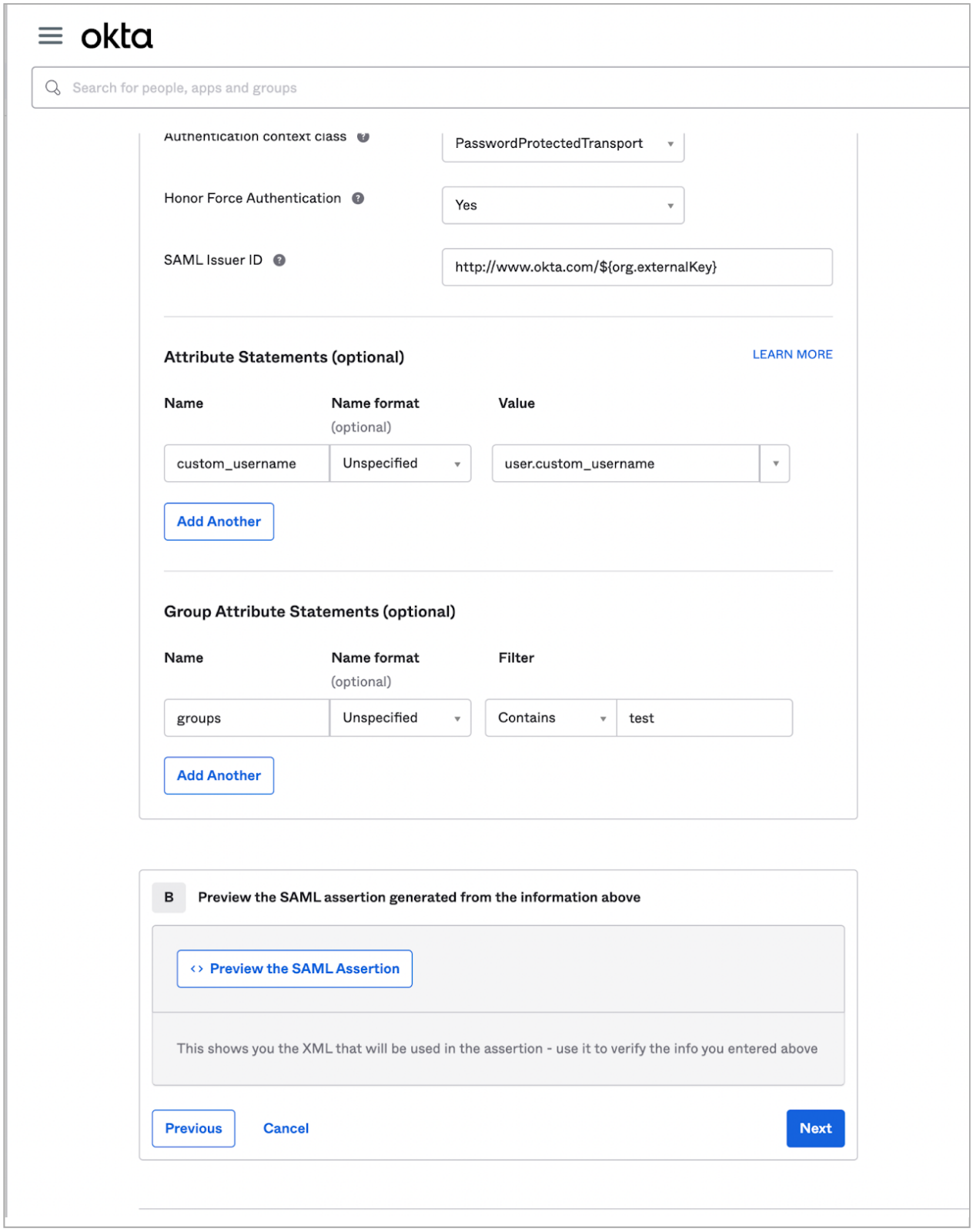


Configuring SAML (Part 2)

2. (Optional) Define custom **Attribute Statements** that Okta will insert in the SAML Assertions shared with Cribl Edge. The only use case that Cribl Edge supports for this setting is creating a custom **Application username**, as described in the previous step.

3. (Optional) 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**.

4. In Part **B**, if you want to see your SAML assertion in XML form, click **Preview the SAML assertion generated from the information above**.



Copy Your App's Metadata to Cribl Edge (Part 3)

1. Click **Next** to open the **Feedback** page, whose fields are self-explanatory, and click **Finish**. Okta takes you to the **Sign On** tab for your newly-created application.

2. Under **SAML Setup** in the right margin, click **View SAML setup instructions** to open the **How to Configure SAML 2.0 ...** page.

3. Copy each value below to its corresponding field in Cribl Edge. Values that you did not configure above will not appear on the page.

SAML APP VALUE	CRIBL EDGE FIELD
Identity Provider Single Sign-On URL	**Single sign-on (SSO) URL**
Identity Provider Single Logout URL	**Single logout (SLO) URL**
Identity Provider Issuer	**Issuer (IDP entity ID)**
X.509 Certificate	**Response validation certificate**

Ignore the **Provide the following IDP metadata to your SP provider** text box. Cribl Edge does not support ingesting metadata in this form.

Advanced Settings

The **Configure SAML** > part **A**, **SAML Settings** page offers these **Advanced Settings**:

Response: This is the SAML response object; it contains an `Assertion` sub-field. Leave this setting as `Signed` (the default), because this is the only alternative that Cribl Edge supports.

Assertion Signature: This applies to the `Assertion` sub-field within the SAML response object. Defaults to `Signed`, but that has no practical effect because Cribl Edge assumes that the whole SAML response object is signed anyway.

Signature Algorithm Select your preferred algorithm for signing the response that the IDP sends to Cribl Edge.

Digest Algorithm: Select your preferred hashing algorithm for the response that the IDP sends to Cribl Edge.

Assertion Encryption: Choose `Encrypted` if you want the IDP to encrypt the response that it sends to Cribl Edge. Setting this to `Encrypted` displays the following three additional options:

- **Encryption Algorithm**: Select your preferred algorithm for encrypting the SAML response.
- **Key Transport Algorithm**: Select your preferred algorithm for encrypting the encryption key itself.
- **Encryption Certificate**: Upload the public key of the Cribl Edge **Response decryption certificate** you selected or created in Cribl Edge.

Signature Certificate: Upload the public key of the Cribl Edge **Request certificate** you selected or created in Cribl Edge.

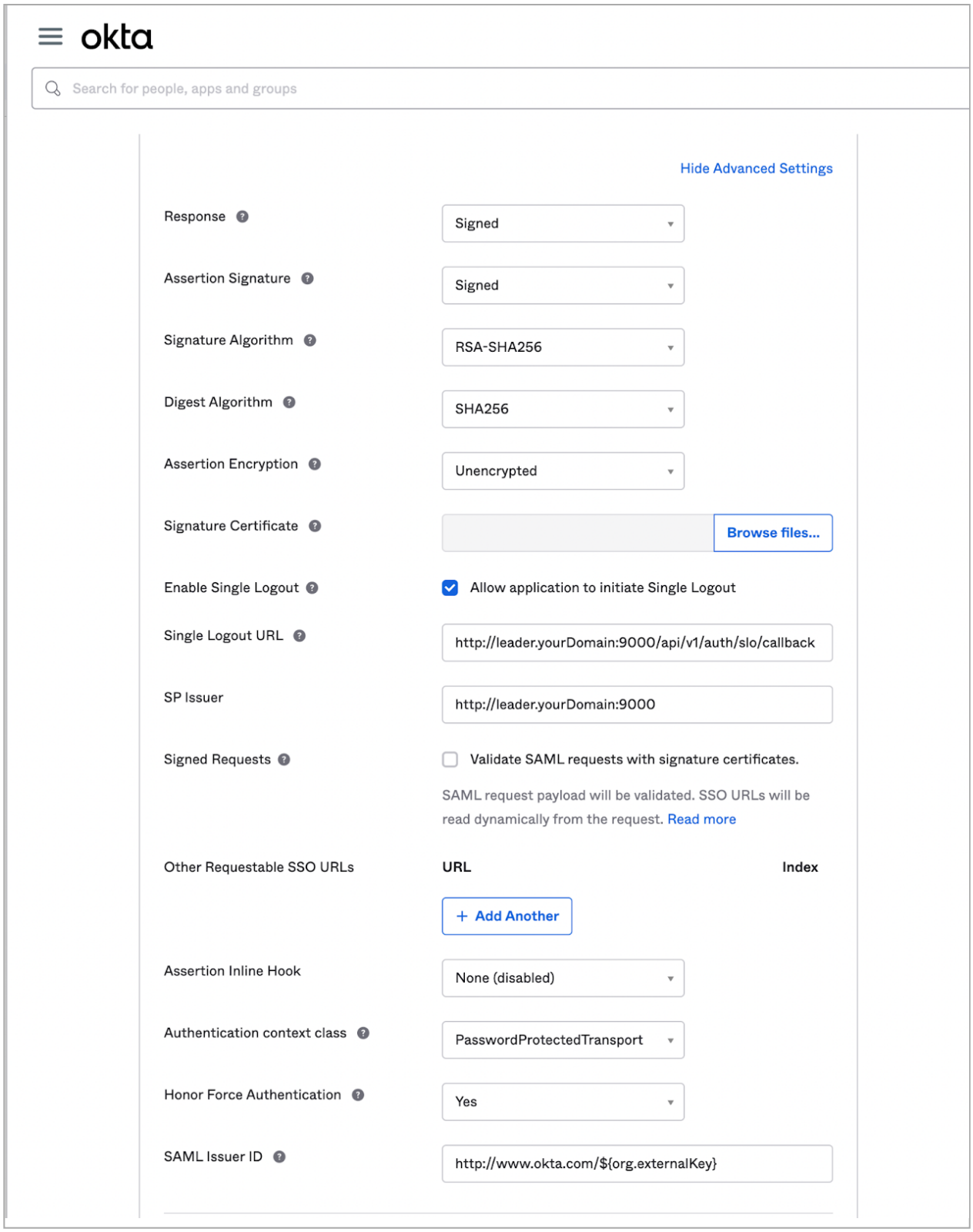
Enable Single Logout: Check the checkbox if you want Cribl Edge to send a logout request to Okta upon logout of the Cribl Edge user. Doing this displays the following two additional options that use values from earlier in these instructions:

- **Single Logout URL**: Enter the value that Cribl Edge provided for **Logout callback URL**.
- **SP Issuer**: Enter the URL that you entered for the **Audience** setting in the Cribl Edge UI.

Signed Requests: Check the checkbox if you want Okta to validate the signatures of SAML requests that come from Cribl Edge.

Other Requestable SSO URLs, **Assertion Inline Hook**, **Authentication context class**, and **Honor Force Authentication**: Ignore these settings (which are not supported by Cribl Edge).

SAML Issuer ID: Enables you to override Okta's default value for **Identity Provider Issuer**.



Okta Advanced Settings

Finish Configuring SAML Auth in Cribl Edge

> This section documents the entire Cribl Edge Authentication UI. If you have been working through the procedures from earlier in this page, you will have completed some of the following steps already.

Navigate to **Settings** > [**Global Settings** >] **Access Management** > **Authentication**. This exposes the following controls.

Audience (SP entity ID): The base URL, e.g.: `https://leader.yourDomain.com:9000` for a distributed environment. For the IDP, this serves as a unique identifier for the SP (i.e., your Cribl Edge instance).

> For distributed environments with a second Leader configured, modify the **Audience** field to point to the load balancer instead of the Leader Node.

Once you enter a URL here, Cribl Edge will automatically populate the following three read-only fields. Copy the values for use in your IDP's UI.

- **Sign-on callback URL**: URL where the SP (Cribl Edge) will consume assertions (i.e., will receive SAML authentication responses from the IDP).
- **Logout callback URL**: URL where the SP (Cribl Edge) will receive SAML logout responses from the IDP.
- **Metadata URL**: URL that exposes a description of the SP (Cribl Edge), including endpoints, certificates, and metadata. Typically, you paste this URL into an **import metadata from URL** or similar field during setup of IDPs that support such an option.

Request certificate: Certificate the SP (Cribl Edge) should use to sign requests. Create a new certificate, or choose one that you imported as described here. Typically pasted into a **Signature Certificate**, **Verification Certificate**, or similar field during IDP setup. Required by some but not all IDPs; for example, when SLO is enabled, Okta requires signed requests.

Issuer (IDP entity ID): Unique ID of the IDP. In the IDP's UI, typically called **Identity Provider Issuer**, **IDP Identifier**, or similar.

Single sign-on (SSO) URL: Endpoint to which authentication requests will be sent (i.e., the IDP's single sign-on service). In the IDP's UI, typically called **Login URL**, **Single Sign-On URL**, or similar.

Single logout (SLO) URL: Endpoint to which logout requests will be sent (i.e., IDP's single logout service). Setting this will enable single logout initiated from the SP (Cribl Edge).

Request binding: Type of binding for SAML requests sent to the identity provider's SSO/SLO services.

Response validation certificate: The certificate that the SP (Cribl Edge) should use to validate signed responses. Provided by the IDP; contains a public key.

> If, in Okta, you have set **Assertion Encryption** to `Encrypted`, then, in Cribl Edge you must both specify a certificate in **Response decryption certificate** below, and, set **Order of encrypting and signing** to `encrypt-then-sign`.

Response decryption certificate: Certificate the SP (Cribl Edge) should use to decrypt encrypted responses. Obtain this from the IDP, where it is typically labeled **Token Encryption**, **Encryption Certificate**, or similar. Create a new certificate, or choose one that you imported as described here.

Order of encrypting and signing: Specify the order in which the IDP signs and encrypts responses, if you've configured it to do that. Requires a **Response decryption certificate** to be set.

Username field: In the SAML response, the element from which Cribl Edge should extract the user identifier.

- Defaults to `NameID`, which is an element in the `Subject` block, for example:

```
<saml:Subject>
    <saml:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified">idp
user</saml:NameID>
    <saml:SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
        <saml:SubjectConfirmationData NotOnOrAfter="2014-02-22T01:20:27.956Z"

Recipient="http://localhost/ExampleServiceProvider/SAML/AssertionConsumerService.as
    </saml:SubjectConfirmation>
</saml:Subject>
```

- Alternatively, if you want to set the username via a custom Attribute when configuring the IDP, you can use the `AttributeStatement` block, for example:

```
<AttributeStatement>
    <Attribute Name="urn:oid:2.5.4.42" FriendlyName="givenName"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:uri">
        <AttributeValue>Dave</AttributeValue>
    </Attribute>
</AttributeStatement>
```

- When you are using a custom Attribute as in the previous bullet point, the value for **Username field** must be what's in the `Name` field of the Attribute – typically an OID-style name (like the example above, where it's `urn:oid:2.5.4.42`). Do **not** use the `FriendlyName`.

Group name field: In the SAML response, the Attribute that specifies the user groups. Defaults to "groups". In the IDP's UI, typically labeled **Group Attribute Statements** or similar.

- When configuring the **Group Attribute Statements** or equivalent in your IDP, view the full Attribute. You should see something like this:

```
<saml2:AttributeStatement xmlns:saml2="urn:oasis:names:tc:SAML:2.0:assertion">
    <saml2:Attribute Name="groups"
NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-format:unspecified">
        <saml2:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="xs:string">
            test-admin
        </saml2:AttributeValue>
        <saml2:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="xs:string">
            test-group
        </saml2:AttributeValue>
    </saml2:Attribute>
</saml2:AttributeStatement>
```

- Note the value for the `Attribute Name`. In the example above, it's `groups`.
- Enter that exact value as the **Group name field** in Cribl Edge.

Allow login as Local User: Toggle to Yes to allow users to log in without SAML (as an alternative, or as a fallback if SAML login fails).

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. See also the explanation of role mapping above. The controls here are:

Default role: Default Cribl Edge Role to assign to all groups not explicitly mapped to a Role.

Mapping: Add a mapping for each external user group that you want to map to one or more Cribl roles. Then enter the group and select the role(s).

See also the explanation of role mapping above.

Role Mapping Example

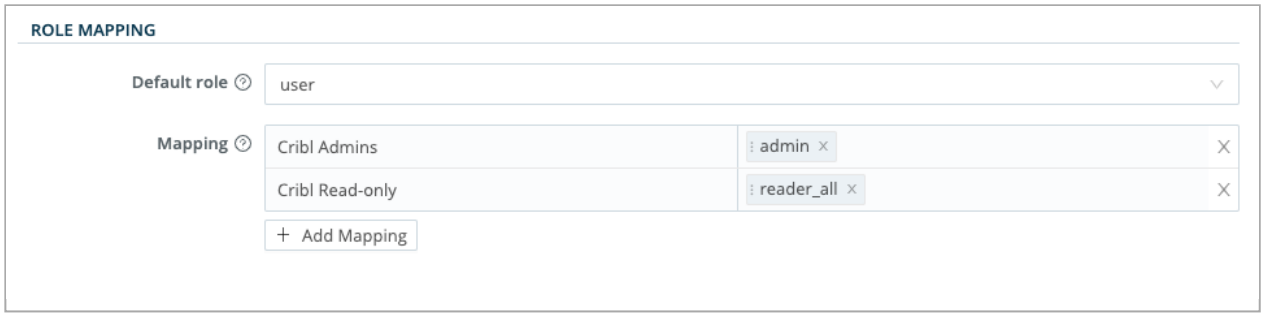
Role mapping UIs will differ from one IDP to another. For this example, we'll look at Okta's.

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, select **Settings** > **Access Management** > **Authentication**.
2. Scroll down to the **ROLE MAPPING** section.

   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).



Role mapping, concluded

Verify that SSO with SAML Is Working

1. Log out of Cribl Edge, and verify that SAML is now an option on the login page.

Logging in with SAML

2. Click **Log in with SAML 2.0**.

3. You should be redirected to your IDP to authenticate yourself.

4. The SSO 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.

16. Troubleshooting

16.1. Known Issues

This page lists known issues affecting Cribl Stream and/or Cribl Edge.

2023-07-27 - All Versions through Current - Recent Actions endpoint returns results for certain unauthorized users [CRIBL-18818]

Problem: The Recent Actions endpoint `GET /api/v1/ui/recentActions` returns more recent actions than intended for non-admin users.

Workaround: None.

Fix: Planned for Cribl Stream 4.2.2.

2023-07-26 – v.4.2.0–4.2.1 – File Monitor Source causes memory leaks and Edge Node crashes [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.

Fix: Planned for Cribl Edge 4.2.2.

2023-07-20 – v.4.2.0–4.2.1 – AppScope Source might reference a nonexistent config [CRIBL-19044]

Problem: The 'sample_config' stock configuration was removed in v.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-premises/customer-managed deployments, clone one of the existing AppScope configurations and rename it to `sample_config`. For Cribl.Cloud customers, there is no workaround yet.

Fix: Version TBD.

2023-07-19 – v.4.2.0–4.2.1 – Users with the Admin Permission cannot view Notifications [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. Organization **Owners** can access these notifications.

Fix: Planned for Cribl Edge 4.2.2.

2023-07-19 – v.4.2.0–4.2.1 – Users with the Admin Permission cannot bootstrap a Worker due to a missing auth token [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.

Fix: Planned for Cribl Edge 4.2.2.

2023-07-19 – v.4.2.0 – Credential encryption errors preventing user access to Cribl Edge Sources and Destinations [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.

Fix: Fixed in Cribl Edge 4.2.1.

2023-07-18 – v.4.2.0–4.2.1 – Migrated Local Users appear in the Members UI with "No Access" [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.

Fix: Version TBD.

2023-07-18 – v.4.2.0–4.2.1 – Product-level Editor Permissions do not allow a commit/deploy action on the Worker Groups/Fleets page [CRIBL-18970]

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.

Fix: Version TBD.

2023-07-18 – v.4.2.0–4.2.1 – Users with the Edge Editor Permission cannot delete secrets under Fleet Settings [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.

Fix: Version TBD.

2023-07-17 – v.4.2.0–4.2.1 – Edge Node Settings not visible when teleporting [CRIBL-18963]

Problem: When you teleport into an Edge Node, you can't view **Node Settings** in the UI.

Workaround: None.

Fix: Planned for Cribl Edge 4.2.2.

2023-07-17 – v.4.2.0–4.2.1 – Users with the Edge Editor Permission cannot delete Sources, Destinations, or Pipelines [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.

Fix: Planned for Cribl Edge 4.2.2.

2023-07-17 – v.4.2.0–4.2.1 – Change from Edge User to Edge Editor can fail to take effect [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.

Fix: Planned for Cribl Edge 4.2.2.

2023-07-17 – v.4.2.0–4.2.1 – Product logins for users with product-level Read Only and Editor Permissions can result in misleading log entries [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.

Fix: Planned for Cribl Edge 4.2.2.

2023-07-16 – v.4.2.0–4.2.1 – After upgrading, Monitoring dashboards are not visible for users with `GroupRead` and

`GroupCollect` Policies [CRIBL-18925]

Problem: Users that were assigned `GroupRead` and `GroupCollect` Policies in Cribl Edge 4.1.x 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.

Fix: Version TBD.

2023-07-15 – v.4.2.0–4.2.1 – Edge users with the Fleet-level Editor Permissions cannot use manual file discovery [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.

Fix: Planned for Cribl Edge 4.2.2.

2023-07-13 – v.4.2.0–4.2.1 – Projects created in v.4.1.x will be deleted when upgrading to Stream v.4.2 [CRIBL-18920]

Problem: Any Projects you created in v.4.1.x will be deleted when you upgrade to v.4.2. You must re-create your Projects after you upgrade. You should also delete and re-create any Subscriptions that carry over during the upgrade.

Workaround: None. All Projects and Subscriptions must be recreated after you upgrade.

Fix: Version TBD.

2023-07-13 – v.4.2.0–4.2.1 – Stream Users are able to access the Monitoring page [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.

Fix: Version TBD.

2023-07-13 – All Versions through Current – When configuring an S3 Collector, JavaScript expressions break the "Auto-populate from" option for the Path field [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.

Fix: Planned for Cribl Edge 4.2.2.

2023-07-12 – v.4.2.0–4.2.1 – No Access to Subfleets for the Edge User Permission [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.

Fix: Planned for Cribl Edge 4.2.2.

2023-07-07 – v.4.1.3-4.2.1 – When configuring an HTTP Source, enabling Source PQ changes the inputId in events [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.

Fix: Version TBD.

2023-07-06 – v.4.2.0–4.2.1 – Default Pack is not visible to Project Editor when a Project is shared with them [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.

Fix: Version TBD.

2023-07-03 – v.4.2.0–4.2.1 – Users with Group-level Read Only Permissions can interact with logging level menus [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.

Fix: Version TBD.

2023-06-29 – All Versions through v.4.1.3 – (Linux) System Metrics Source does not emit "Per interface" metrics when set to "All" [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`.

Fix: In Cribl Edge 4.2.

2023-06-28 – v.4.2.0–4.2.1 – Project editor can't delete Pipelines via Options menu [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**.

Fix: Planned for Cribl Edge 4.2.2.

2023-06-26 – All Versions through v.4.1.3 – High CPU usage in File Monitor Source's Manual mode [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.

Fix: In Cribl Edge 4.2.

2023-06-26 – All Versions through Current – Fleet Upgrade Errors [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.

Fix: Version TBD.

2023-06-21 – v.3.5.2–4.1.3 – Using the Rename Function gives unexpected results due to skipped internal fields [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.

Fix: In Cribl Edge 4.2.

2023-06-20 – All versions through 4.2.2 – HTTP Destinations sometimes send oversized payloads [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.

Fix: Planned for Cribl Edge 4.3.

2023-06-15 – v.4.1.3 – Amazon CloudWatch Destinations log error while flushing events older than 24 hours [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.`

Fix: In Cribl Edge 4.2.

2023-06-14 – v.4.1.2–4.1.3 – The Office 365 Activity Source misses events [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.

Fix: In Cribl Edge 4.2.

2023-06-08 – v.4.1.2 – AES-256-GCM security vulnerability [CRIBL-18038]

Problem: The AES-256-GCM encryption option introduced in v.4.1.2 included a security vulnerability.

Workaround: Do not use this option in v.4.1.2.

Fix: In Cribl Edge 4.1.3.

2023-06-01 – v.4.0.0–4.1.3 – GroupEdit is not able to commit changes [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.

Fix: In Cribl Edge 4.2.

2023-06-01 – v.4.1.0–4.1.2 – Kubernetes Logs Event Breaker Incorrectly Breaks Events [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.

Fix: In Cribl Edge 4.1.3.

2023-05-27 – v.4.1.1–4.1.2 – Perf bug: RPC consuming excessive CPU compiling expressions [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.

Fix: In Cribl Edge 4.1.3.

2023-05-26 v.4.1.2–4.1.3 – Edge Nodes don't update from Cribl Edge Leader [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.

Fix: In Cribl Edge 4.2.

2023-05-19 – v.4.1.2 – CrowdStrike Destination's "LogScale endpoint" field hidden from UI [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.

Fix: In Cribl Edge 4.1.3.

2023-05-16 – v.4.1.1–4.1.2 – Expanding Status of Output Router Destination triggers error [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.

Fix: In Cribl Edge 4.1.3.

2023-05-12 – v.4.0–4.1.3 – Windows Event Forwarder Source causes duplicate events when experiencing backpressure [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.

Fix: In Cribl Edge 4.2.

2023-05-12 – v.4.0–4.1.2 – The Kubernetes Logs Source drops events [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.

Fix: In Cribl Edge 4.1.3.

2023-05-05 – v.4.1.1 – S3 ingestion slows after upgrade [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.

Fix: In Cribl Edge 4.1.2.

2023-04-27 – v.4.1.1 – Edge Nodes integrated with `systemd/initd` can fail upon reconfig [CRIBL-17264]

Problem: Where 4.1.1 on-prem or hybrid 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 v.4.1.1 or roll back to your last stable version. To run v.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>`.

Fix: In Cribl Edge 4.1.2.

2023-04-14 – v.2.2–4.1.x – Requests from Office 365 Message Trace Source failed intermittently [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.

Fix: Fixed by Microsoft (not a Cribl problem).

2023-04-13 – v.4.1.1 – Monitoring incorrectly shows metrics for disconnected Subscriptions [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.

Fix: In Cribl Edge 4.1.2.

2023-04-03 – v.4.1.0 – Windows Event Logs Source should have a configurable Max Event Size [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.

Fix: In Cribl Edge 4.1.1.

2023-03-31 – v.4.1.0-4.1.1 – Cribl.Cloud API Credential's Roles aren't honored on tokens [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.

Fix: In Cribl Edge 4.1.2.

2023-03-30 v.4.1.0 – Splunk Load Balanced Destination incorrectly re-creates all connections during DNS resolution [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.`

Fix: In Cribl Edge 4.1.1.

2023-03-28 – v.4.1.0 – Add Kubernetes and Docker modals in the UI aren't checking the `CRIBL_BOOTSTRAP_HOST` environment variable [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.

Fix: In Cribl Edge 4.1.1.

2023-03-23 – v.4.1.0 – Leader's config deployments to pre-4.1 Workers silently fail [CRIBL-16339]

Problem: A 4.1.x Leader requires Workers to also be upgraded to – v.4.1.x. 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 on-prem or hybrid Workers, upgrade all Fleets to 4.1.x before you deploy to them.

Fix: In Cribl Edge 4.1.1.

2023-03-22 – v.4.1.0-4.1.1 – Cribl Stream Database Collector cannot connect with SQL Server using AD authentication [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 v.4.0.4 restores the Database Collector's ability to connect using AD authentication.

Fix: In Cribl Stream 4.1.2.

2023-03-21 – v.4.1.0-4.2.1 – Unable to bind `CRIBL_DIST_MASTER_URL` to an IPv6 address [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
```

Fix: Planned for Cribl Edge 4.2.2.

2023-03-20 – v.3.5.0–4.1.0 – System Activity drawer on Windows Edge Nodes displays no data [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.

Fix: In Cribl Edge 4.1.1.

2023-03-20 – v.4.1.0-4.2.1 – If a deployment fails, Workers will not automatically revert to the previous version [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.

Fix: Planned for Cribl Edge 4.2.2.

2023-03-15 – v.4.1.0 – QuickConnect Pipeline Settings button fails [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.

Fix: In Cribl Edge 4.1.1.

2023-03-14 v.4.0.0–4.1.0 – Duplicate data with Event Hubs [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**.

Fix: In Cribl Edge 4.1.1.

2023-03-01 – v.4.0.4 – Kafka-based Sources are omitting the `_time` field [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.

Fix: In Cribl Edge 4.1.

2023-02-24 – v.4.1.0-4.2.1 – The Chain Function has a 10% impact on performance [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.

Fix: Planned for Cribl Edge 4.2.2.

2023-02-23 – v.4.0.x – DNS Resolution reconnects Cribl TCP connections every time [CRIBL-15363]

Problem: All Cribl TCP connections will reconnect every time DNS Resolution occurs, even when reconnection is not necessary.

Fix: In Cribl Edge 4.1.

2023-02-23 – v.4.0.4–4.1.2 – Enable Automatic Upgrades deletes remote repo's Git Settings [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**.

Fix: In Cribl Edge 4.1.3.

2023-02-22 – v.4.0.4–4.1.0 – Sending data to an inactive Kafka, Confluent Cloud, or Azure Event Hubs Destination triggers an endless series of log errors [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.

Fix: In Cribl Edge 4.1.1.

2023-02-20 – v.4.0.x–4.1.0 – Logging-level changes do not take effect for services [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.

Fix: In Cribl Edge 4.1.1.

2023-02-20 – v.4.0.3–4.1.0 – On-prem Edge Nodes automatically upgrade, ignoring Leader settings [CRIBL-15367]

Problem: Upgrading a Leader Node causes its 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 Cloud Edge Nodes.

Workaround: 1. Toggle the UI's **Enable automatic upgrades** slider 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`.

Fix: In Cribl Edge 4.1.1.

2023-02-14 – v.4.0.0–4.0.4 – Pipeline is deselected in QuickConnect after modification [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.

Fix: In Cribl Edge 4.1.

2023-02-14 – v.3.5.0 through Current – Cannot clear Messages drawer while in GitOps Push mode [CRIBL-15239]

Problem: When in GitOps `Push` mode, you cannot clear messages from the **Messages** drawer. Cribl Edge returns a `Forbidden` error.

Fix: Version TBD.

2023-02-06 – v.4.0.x–4.1.0 – Event Breaker timestamp extraction fails after configured time [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.

Fix: In Cribl Edge 4.1.1.

2023-02-05 – v.3.0.3–4.0.4 – `CRIBL_DIST_WORKER_PROXY` env var is ignored when Leader Node's Master URL is set via `instance.yml` [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`.

Fix: In Cribl Edge 4.1.

2023-02-03 – v.4.0.0–4.0.4 – Metrics blocklist can’t be changed on a global level [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.

Fix: In Cribl Edge 4.1.

2023-01-26 – v.3.5.0–4.0.3 – Preview Full > Send Out does not capture events [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.

Fix: In Cribl Edge 4.0.4.

2023-01-25 – v.3.5.x through Current – Cascading problems when Windows Event Logs Source collects from `ForwardedEvents` channel [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.

Fix: Version TBD.

2023-01-17 – v.4.0.x–4.0.4 – GitOps Push/read-only confirmation banner has dead link [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`.

Fix: In Cribl Edge 4.1.

2023-01-12 – v.3.5.x through Current – Recently added Worker Nodes fail to appear on the Monitoring page [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.

Fix: Version TBD.

2023-01-11 – Multiple versions through 4.1.x – Kafka-based Sources' rebalancing is logged with exaggerated severity [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.

2023-01-10 – v.3.5.x through Current – Configuration updates not shared between primary and standby Leaders [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.

Fix: In – v.4.0.3, 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. Version TBD for automatic synchronization between Leaders' hosts.

2023-01-02 – v.4.0.0–4.0.2 – Google Cloud Chronicle Destination drops changes in distributed deployments with custom log types [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.

Fix: In Cribl Edge 4.0.3.

2022-12-15 – v.3.5.4–4.0.3 – Changes to a Lookup table on the Leader don't always propagate to Edge Nodes [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.

Fix: In Cribl Edge 4.0.4.

2022-12-13 – v.4.0.0 through Current – Default commit message missing for non-admin users [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.

Fix: Version TBD.

2022-12-09 – v.4.0.1–4.0.2 – Users assigned the `owner_all` role cannot perform commits [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 }`.

Fix: In Cribl Edge 4.0.3.

2022-12-08 – v.4.0.1–4.0.2 – Landing page not displaying system metrics from Edge Nodes when teleporting from the Leader [CRIBL-14169]

Problem: When you teleport from a Leader running v.4.0.1 or 4.0.2 to an Edge Node that's running v.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 v.4.0.2 or higher.

Fix: In Cribl Edge 4.0.3.

2022-12-07 – v.4.0.1 – Google Cloud Pub/Sub Source and Destination can break on field validation [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.

Fix: Validation is rolled back in Cribl Edge 4.0.2.

2022-12-01 – v.4.0.2 – CrowdStrike FDR Source needs 6-hour visibility timeout [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.

Fix: In Cribl Edge 4.0.3.

2022-12-01 – v.4.0.0 – Amazon S3 Source stops receiving data after upgrade [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 v.4.0.0 (or temporarily roll back to v.3.5.4).

Fix: In Cribl Edge 4.0.1.

2022-11-30 – v.4.0.2 – QuickConnect-configured Sources' misleading "Enabled" status while disconnected [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.

Fix: In Cribl Stream 4.0.3.

2022-11-28 v.4.0–4.0.2 – “Unknown config version” error can appear when deploying changes [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 v.1.9.1 or later.

Fix: In Cribl Edge 4.0.3.

2022-11-28 – v.4.1.0 through Current – Diag bundles might fail to download from teleported Edge Nodes [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.

Fix: Version TBD.

2022-11-23 – All versions through 4.1.0 – Edge Nodes show incorrect configurations after upgrade or commit/deploy from Leader [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.

Fix: In Cribl Edge 4.1.1.

2022-11-23 – v.4.0.0 – Teleported Edge Nodes don't display Distributed Settings or Diagnostics Settings [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.

Fix: In Cribl Edge 4.0.1.

2022-11-22 – v.3.2.0–4.0.0 – Chain Function creates extra Pipelines and Functions during initialization [CRIBL-13860]

Problem: Under certain circumstances, when initializing, the Chain Function created multiple unneeded Pipelines and Functions.

Fix: In Cribl Edge 4.0.1.

2022-11-21 – v.4.0.0 – Metrics service unavailable [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 file to `/etc/tmpfiles.d` 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 Cloud.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.

2022-11-18 – v.4.0.0–4.0.3 – License usage visible only for current day [CRIBL-13811]

Problem: In the listed versions, selecting **Monitoring** > **System** > **License** displays usage only for the current day.

Fix: In Cribl Edge v.4.0.4.

2022-11-16 – versions 3.4.0–4.0.4 – Source PQ in Smart mode can trigger excessive memory usage and OOM failures [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.

Fix: In Cribl Edge 4.1.

2022-11-14 – version 4.0.0 – `Forbidden` error banners mistakenly displayed to non-admin users [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.

Fix: In Cribl Edge 4.0.1.

2022-11-13 – v.4.0.0–4.0.2 – Edge/Kubernetes Logs Source repeats logs due to incorrect timestamp [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.

Fix: In Cribl Edge 4.0.3.

2022-11-08 – v.4.0.0 – Cribl Edge honeycomb display doesn't render null values [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`.

Fix: In Cribl Edge 4.0.1.

2022-11-08 – v.4.0.0 – Broken in-app doc links [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.

Fix: In Cribl Edge 4.0.1.

2022-11-07 – v.4.0.0 – Spurious “Secret decrypt failed with error" log messages appear after upgrade [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.

Fix: In Cribl Edge 4.0.1.

2022-11-07 – v.4.0.0 – Upgrade status shown on Group Upgrade page doesn't match Global Settings [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.

Fix: In Cribl Edge 4.0.1.

2022-11-04 – v.3.5.1–4.0.4 – Avg Thruput (Bytes Per Second) not displayed for internal logs or metrics [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.

Fix: In Cribl Edge 4.1.

2022-11-03 – v.4.0.0 – Sorting by column on Manage > Edge Node page results in blank list [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.

Fix: In Cribl Edge 4.0.1.

2022-11-01 – All versions through Current – Source PQ in Smart mode could cause premature backpressure [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.

Fix: Version TBD.

2022-11-01 – v.4.0.0 through Current – Default AppScope Config file can't be restored when in use [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.

Fix: Version TBD.

2022–10-20 – v.4.0.0 through Current – Pipelines with Clone and other Functions can show inconsistent total processing times [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**.

Fix: Version TBD.

2022-11-03 – v.4.0.0 – Git settings don't immediately populate Commit modals [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.

Fix: In Cribl Edge 4.0.1.

2022-10-12 – All versions through 3.5.4 – Event preview can hide some fields [CRIBL-12914]

Problem: Preview panes can hide some fields at the bottom of each event.

Workaround: Resize your browser window to give the preview pane more depth.

Fix: In Cribl Edge 4.0.

2022-10-11 – All versions through Current – GitOps Push mode doesn't support ad hoc Collection jobs [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.

Fix: Version TBD.

2022-10-04 – All versions through 3.5.4 – File Monitor Source fails with high-volume logs and file rotation [CRIBL-12762]

Problem: The File Monitor Source might stop reading logs from an open file, if the log rotation service renames that file.

Workaround: Restart the File Monitor Source.

Fix: In Cribl Edge 4.0.

2022-10-03 – All versions through 4.0.4 – New Help drawers open under pinned Help drawers [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.

Fix: In Cribl Edge 4.1.

2022–09-26 – All versions through 4.0.0 – Manage Groups/Fleets pages count Packs in Pipeline totals [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.

Fix: In Cribl Edge 4.0.1.

2022-09-19 – v.4.0.0 – Subfleet search doesn't return parent Fleets [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.

Fix: In Cribl Edge 4.0.1.

2022-09-06 – v.3.3.0–3.5.4 – Load-balanced Destinations with PQ enabled: Buffer flush errors and memory leaks [CRIBL-12203]

Problem: Destination logs might show errors of the form: `Attempted to flush previously flushed buffer token`. No data has been lost, but this will lead to a memory leak until the Edge Node is restarted. This is most likely to occur when there are more available downstream hosts than a **Max connections** limit configured on the Destination.

Workaround: To prevent the bug from occurring, set **Max connections** to the default `0` (enabling unlimited connections). This will ensure that all discovered hosts enter the connection pool. If you encounter the bug, restarting the affected Edge Nodes (or letting an OOM the killer do so) will clear the memory leak.

Fix: In Cribl Stream 4.0.

2022-08-29 – All versions through 3.5.4 – Sources reject connections on ACME certs with no Subject [CRIBL-12097]

Problem: With multiple Sources, if you configure TLS mutual authentication using an ACME certificate with an empty `Subject` field, Cribl Edge will reject connections – even though RFC 6125 allows for `Subject` to be empty. This affects: Splunk Sources, Cribl internal Sources, Syslog, TCP, TCP JSON, Metrics, HTTP/S, Amazon Firehose, Elasticsearch API, DataDog, Grafana, Loki, Prometheus, and Windows Event Forwarder.

Workaround: Enter any value in the certificate's `CN` field. Any entry will cause CCribl Edge to match on the certificate's SAN (`subjectAltName`) extension.

Fix: In Cribl Edge 4.0.

2022-08-20 – v.3.2.0–4.0.3 – Deleting or modifying default Mapping Rule reclassifies Cribl-managed Cloud Workers as hybrid

Workers [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 Cloud Workers as 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.

Fix: In Cribl Edge 4.0.4.

2022-08-19 – v.3.5.1–3.5.4 – Using GitOps environment variables crashes the Leader [CRIBL-11972]

Problem: Where deployments rely on `CRIBL_GIT_*` environment variables, the Leader might crash on startup.

Workaround Restart the Leader, disabling the environment variables if necessary.

Fix: In Cribl Edge 4.0.

2022-08-27 – All versions through 4.1.3 – Azure Event Hubs Destination with PQ drops events when inbound data is interrupted [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.

Fix: In Cribl Edge 4.2.

2022-08-25 – Versions 3.4.1 through Current – Splunk Load Balanced Destination degradation with acks enabled [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**.

Fix: Planned for Cribl Edge 4.2.2.

2022-08-16 – All versions through Current – Splunk HEC shows higher outbound data volume than other Splunk Destinations

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.

Fix: Version TBD.

2022-08-16 – v.3.1.2–3.5.2 – Auto timestamping randomly begins using current time [CRIBL-11925]

Problem: After a random interval, the auto timestamping feature might insert the current time into the `_time` field instead of using the timestamp from the event.

Workarounds: Restart the Stream instances to temporarily remedy the problem, or enable an Event Breaker rule's Manual Timestamp Format option.

Fix: In Cribl Edge v.3.5.3.

2022-08-10 – v.3.5.1-3.5.2 – Changing Notification target type crashes New Target modal [CRIBL-11848]

Problem: Changing the **Target Type** while configuring a new Notification target crashes the **New Target** modal.

Workaround: Set the **Target type** once only while the modal is open.

Fix: In Cribl Edge 3.5.3.

2022-08-08 – v.3.x through Current – `cidr-matcher` does not support IPv6 addresses [CRIBL-11767]

Problem: The version of `cidr-matcher` supported by Cribl only supports IPv4 CIDR matching.

Fix: Version TBD.

2022-08-03 – v.3.5.0–3.5.1 – Cribl HTTP and Cribl TCP Sources are not enabled on Cribl.Cloud [SAAS-1905]

Problem: Cribl.Cloud's **Network Settings** >: **Data Sources** tab lists **Ingest Address** ports for the new Cribl HTTP and Cribl TCP Sources. However, these ports are not yet enabled.

Workaround: If you need these ports enabled before the next maintenance release, contact Cribl Support or your Solutions Engineer.

Fix: In Cribl Edge 3.5.2.

2022-08-02 – v.3.3.0–3.5.1 – Managed Edge Nodes mistakenly display "Unregistered" button [CRIBL-11652]

Problem: Managed Edge instances erroneously display an **Unregistered** button. Managed Edge Nodes cannot be registered, because they pull their registration/licensing information from the Leader. So this button should not appear.

Workaround: Ignore the scary button. Your managed Edge Node has inherited its Leader's registration.

Fix: In Cribl Edge 3.5.2.

2022-08-02 – v.3.x through 3.5.1 – Cribl.Cloud displays unsupported Sharing settings [SAAS-1899]

Problem: On Cribl.Cloud, Cribl Edge's **General Settings** > **Upgrade & Share Settings** display a **Sharing and live help** toggle, even though you cannot turn off telemetry on Cribl- or customer-managed (hybrid) Cloud Workers.

Workaround: Ignore this toggle. The UI controls enable you to slide the toggle to `No` and save the result, but this will have no effect.

Fix: In Cribl Edge 3.5.2.

2022-07-21 – v.3.5.0–3.5.2 – Job Inspector shows higher Bytes In than Monitoring dashboards [CRIBL-11482]

Problem: The Job Inspector sometimes displays a higher byte count than other Monitoring dashboards display for the same collection job. The Job Inspector overstates the throughput because it disregards any Event Breakers and Filter expressions applied between initial collection and Pipelines.

Workaround: Rely on the Monitoring metrics to judge accurate data flow through Pipelines to downstream services.

Fix: In Cribl Edge 3.5.3.

2022-07-21 – v.3.4.0–3.5.1 – Data loss with Source-side persistent queueing enabled [CRIBL-11478]

Problem: With Source-side PQ enabled, events got stuck and did not process through Pipelines. The root cause was a Rename Function that used wildcards to rename required internal fields (specifically, `__pqSliceId` and `__offset`).

Workaround: Where Rename Functions potentially rename internal fields (especially via **Rename expressions** that include wildcards), either disable PQ on Sources that feed the parent Pipeline, or test and narrow the expression to affect only the desired fields.

Fix: In Cribl Edge 3.5.2.

2022-07-21 – v3.5.1-3.5.4 – Fake Pack appears after upgrading to v.3.5.1. [CRIBL-11481]

Problem: After an upgrade to v.3.5.1., a new Pack appears on the **Manage Packs** page. This Pack has no **Display Name** and no contents.

Workaround: Delete the ghost Pack.

Fix: Not observed as of Cribl Edge 4.0.

2022-07-20 – v3.5.0–3.5.1 – Manage Edge Nodes page fails to lazy-load Workers when scrolled [CRIBL-11474]

Problem: Scrolling the **Manage Edge Nodes** page fails to load more than 50 Edge Nodes.

Fix: In Cribl Edge 3.5.2.

2022-07-20 – v3.5.0–4.0.4 – Cribl Edge/Windows: Upgrading ignores existing mode (etc.) settings [CRIBL-11467]

Problem: When re-running 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.

Fix: In Cribl Edge 4.1.

2022-07-19 – v.3.5.0–3.5.1 – Cribl Edge/Windows: Syslog Source disallows UDP binding [CRIBL-11439]

Problem: On Cribl Edge/Windows, attempting to bind the Syslog Source to a UDP port might trigger an error of the form: `bind EADDRINUSE 0.0.0.0:<port number>`. The reason is that Edge currently does not fully support the `socket.bind` on Windows.

Fix: In Cribl Stream 3.5.2.

2022-07-13 – v.3.5.0–3.5.1 – Cribl Edge: Fleet > List View tab doesn't correctly filter Edge Nodes [CRIBL-11183]

Problem: A Fleet Home page's **List View** tab doesn't correctly filter Edge Nodes. This tab displays all the Edge Nodes across all Fleets, instead of only those assigned to the Fleet.

Workaround: The **Map View** page is filtered correctly.

Fix: In Cribl Edge 3.5.2.

2022-07-04 – All versions through v.3.5.3 – No extended characters in Publish Metrics Function > Event field name field [CRIBL-8968, CRIBL-10984]

Problem: A Publish Metrics Function's **Event field name** values should contain only letters, numbers, underscores (_), and . characters (to separate names in nested structures). Using other characters will cause the parent Pipeline to stop sending events. Cribl Edge 3.5.0 and above check for valid characters when

you save the Function's configuration, but in prior versions, the invalid config will save without an error message – the failure will happen at runtime, with errors echoed only in the logs.

Workaround: Ensure that **Event field name** values contain no extended characters.

Fix: In Cribl Edge 3.5.4.

2022-07-01 – All versions through 3.5.0 – S3-based Sources' unread Region triggers auth errors [CRIBL-10981]

Problem: On Amazon S3–based Sources, if you concatenate the AWS Region into the **Queue** field's URL or ARN, the SQS client is correctly configured to that region, but the S3 client is not. You will see authentication errors of the form: `The AWS Access Key ID you provided does not exist in our records.`

Workaround: Explicitly set the **Region** drop-down, even if the URL/ARN already contains the same Region.

Fix: In Cribl Edge 3.5.1.

2022-06-30 – All versions through Current – Persistent queue with compression requires oversize max queue limit [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.

Fix: Version TBD.

2022-06-29 – v.3.4.2–3.5.1 – Can't install a new Cribl Edge instance as a named user [CRIBL-10907, CRIBL-11457]

Problem: Bootstrapping a new Leader or Worker with a command of this form fails, with no systemd service created: `cribl boot-start enable -u <username>`

The symptom is an error of the form:
`error: found user=0 as owner for path=/opt/cribl/local, expected uid=1001.` This does not affect upgrades to existing instances.

Workaround: Issue the `[sudo] chown -R <username>: /opt/cribl` command a second time. Then reattempt the `cribl boot-start` command.

Fix: In Cribl Edge 3.5.2.

2022-06-28 – v.3.5.x through Current – Cribl Edge/Windows Limitation: Cannot upgrade Edge Nodes from the Leader's UI [CRIBL-10870]

Problem: Cribl Edge/Windows does not support upgrading Edge Nodes via the Leader's UI.

Workaround: Re-run the Windows installer on each Edge Node, specifying the same installation options/parameters you used when installing the preceding version.

Fix: Planned for Cribl Edge 4.2.2.

2022-06-27 – v.3.3.0–3.5.1 – Edge vs. Stream conflict in `/tmp` directory [CRIBL-10806]

Problem: If you install Cribl Edge and Cribl Stream on the same machine or VM – e.g., running Edge as `root` and Stream as a `cribl` user – both services attempt to use the `tmp/cribl_temp` subdirectory. Starting Stream after Edge will throw an error of the form: `EPERM: operation not permitted, rmdir '/tmp/cribl_temp'`. Starting Edge after Stream will change the folder's ownership, blocking subsequent restarts of Stream.

Workaround: For either Cribl Edge or Cribl Stream, set the `CRIBL_TMP_DIR` variable to a separate base subdirectory like `/tmp/stream/` or `/tmp/edge/` before starting the app.

Fix: Cribl Edge 3.5.2 and later set separate `tmp/` subdirectories by default.

2022-06-24 – v.3.5.0 – Leader takes excessive time to report healthy status [CRIBL-10768, CRIBL-11127]

Problem: Upgrading from v.3.4.x to v.3.5.0 tripled the latency before some Leaders reported healthy status. The presumed cause was that load time increased due to an accumulation of persistent metrics, which progressively increased with more Worker Nodes and more uptime.

Workaround: If upgrading is not possible or insufficient, move the existing metrics subdirectory away from `$CRIBL_HOME/state/metrics/`. Cribl Edge will re-create that as an empty subdirectory, and begin accruing fresh metrics there.

Fix: In Cribl Stream 3.5.1.

2022-06-23 – v.3.5.0 – Logs not viewable at Worker Group level [CRIBL-10665]

Problem: If you select **Monitoring** > **Logs**, and then select a Worker Group from the drop-down at upper left, you might see no logs at the Group level. Instead, you will see an error banner of the form: `ENOENT: no such file or directory, scandir '/opt/cribl_data/failover/log/group/<group-name>'`

Workaround: From the drop-down at upper left, select individual Worker Nodes to view their logs.

Fix: In Cribl Edge 3.5.1.

2022-06-09 – v.4.0.x–4.1.0 – Edge Settings mistakenly display Process Count controls [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.

Fix: In Cribl Edge 4.1.1.

2022-06-01 – v.3.4.2-3.5.0 – Bootstrapping fails to create SystemD service file, or starts wrong service [CRIBL-10250]

Problem: Bootstrapping a 3.4.2 Worker fails. The terminal displays no `Enabled Cribl to be managed by ...` confirmation message.

Workaround: Explicitly run the `boot-start` CLI command.

Fix: In Cribl Edge 3.5.1.

2022-05-31 – v.3.4.2 – Cloning/loading Groups page breaks with `Config Helper service is not available...` error [CRIBL-10228, CRIBL-10229]

Problem: After cloning a Group containing at least one Pack, the UI hangs with a spinning pinwheel and an error banner reading `Config Helper service is not available...`. The root cause is that cloning the Group failed to copy over the Pack, leaving the new Group with broken references to it.

Workaround: Use the filesystem to manually copy missing Packs from the original Group to the cloned Group. If these Packs' contents are not needed in the new Group, it's sufficient to just create the parallel subdirectory with: `mkdir $CRIBL_HOME/groups/$destGroup/default/$packName`.

Fix: In Cribl Edge 3.5.

2022-05-19 – v.3.3.0–3.5.1 – System Metrics Source skips filesystem events on LVM volumes [CRIBL-10092]

Problem: When monitoring LVM logical volumes, the System Metrics Source omits `node_filesystem_*` events.

Fix: In Cribl Edge 3.5.2.

2022-05-06 – v.3.5.0-3.5.4 – Running ~1,000 Edge Nodes crashes Leader [CRIBL-9859]

Problem: Attempting to bring up approximately 1,000 Edge Nodes (even in small batches) can crash the Leader.

Workaround: To support up to 1,000 Nodes, apply this Node.js setting:
`export NODE_OPTIONS=--max-old-space-size=8192`

Fix: In Cribl Edge 4.0.

2022-04-28 – v.3.4.0–3.4.1 – REST Collector Misinterprets `.` character in Response Attributes [CRIBL-9708]

Problem: Where a REST Collector's Response Attributes contain a `.` character, it misinterprets this as an attribute nested within an object, not a literal character.

Workaround: If possible, use other Response Attributes to fetch the next page. Otherwise, skip this version, or downgrade to a known well-behaved version.

Fix: In Cribl Edge 3.4.2.

2022-04-27 – v.3.4.1 – Git Changes drop-down returns `Invalid Date` [CRIBL-9698]

Problem: After opening the left nav's **Changes** fly-out, the **Version** drop-down can display `Invalid Date` instead of commits' dates.

Workaround: Upgrade your installed `git` client to v.2.1.1 or newer; or skip this Cribl Edge version.

Fix: In Cribl Edge 3.4.2.

2022-04-26 – v.3.4.1 – HTTP 500 error when git is absent [CRIBL-9684]

Problem: A `500–Internal Server Error` error banner can indicate that `git` is not installed on Cribl Edge's host.

Workaround: Install `git` on the Leader or single instance. On affected Edge Nodes, install `git`, followed by a `git init` and an empty push to a new `master` branch. A simpler workaround for Edge Nodes is to just enable remote access (Stream, Edge) from the Leader.

Fix: In Cribl Edge 3.4.2.

2022-04-22 – v.3.3.0-3.4.1 – Datadog Agent API key ignored when forwarding metrics to Datadog [CRIBL-9633]

Problem: On a Datadog Destination, when **Allow API key from events** is set to `Yes`, the API key from a Datadog Agent Source is not correctly emitted with metric events sent to Datadog. Instead, Cribl Edge simply sends the static **API key** configured in the Destination. (This affects metrics events only; other data types correctly forward the event's API key.)

Workaround: Set **Allow API key from events**. to `No`. Configure the desired API key in the Destination's static **API key** field. (If you need multiple API keys, can create multiple Destinations, and route events to each based on the `__agent_api_key` field value. This value contains the Agent API key per event.)

Fix: In Cribl Edge 3.4.2.

2022-04-21 – v.3.4.1 – Failed restart, `Uncaught (in promise)` error, after changing Authentication settings [CRIBL-9614]

Problem: After changing Authentication settings, or configuring an external auth provider, Cribl Edge might fail to restart. Indications are errors of the form: `Uncaught (in promise) TypeError: Cannot read properties of undefined (reading 'workerRemoteAccess')`, or: `Uncaught (in promise) TypeError: l.api is undefined.`

Workaround: Directly edit the `cribl.yml` file's `auth:` section.

Fix: In Cribl Edge 3.4.2.

2022-04-21 – All versions – Cribl.Cloud login page distorted on iPad [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.

Fix: No planned fix.

2022-04-19 – v.3.3.0–3.4.2 – Spurious buffer token flush error in TCP-based Destinations with PQ enabled [CRIBL-9565]

Problem: This error message can mistakenly appear in logs: `Attempted to flush previously flushed buffer token`. You can ignore it on TCP-based, load-balanced Destinations (Splunk Load Balanced, TCP JSON, Syslog/TCP, and Cribl Stream) with persistent queueing enabled.

Fix: In Cribl Edge 3.5.

2022-04-18 – All versions through Current – Upgrade Version List is Limited to Latest Version [CRIBL-9568]

Problem: The **Choose Version** drop-down list displays the latest version only. This creates an issue when you want to incrementally upgrade between prior versions.

Fix: Version TBD.

2022-04-18 – v.3.3.1–4.0.0 – Lookup Functions intermittently fail [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.

Fix: In Cribl Edge 4.0.1.

2022-04-15 – All versions through 3.4.1 – Git permission errors [CRIBL-9530]

Problem: Multiple Linux distro's have backported a permissions-restriction patch from `git-2.35.1` to earlier versions, notably including Ubuntu 20.04 LTS. If you see errors of the form `fatal: unsafe repository ('/opt/cribl' is owned by someone else)` on the command line or in the Cribl Edge UI, this indicates an ownership mismatch (current user versus file base) on the directory corresponding to `$CRIBL_HOME` or `$CRIBL_VOLUME_DIR`.

Workaround: Use `chown` to recursively set permissions on all files in `/opt/cribl/` (or in or `$CRIBL_VOLUME_DIR`'s target directory) to match the user running Cribl Edge.

Fix: In Cribl Edge 3.4.2.

2022-04-15 – v.3.4.0 – Workers report incomplete metrics [CRIBL-9524]

Problem: After upgrading to v.3.4.0, Worker Nodes failed to report all metrics. Missing metrics were logged with a warning of the form: `failed to report metrics`, and with a reason of the form: `Cannot read property 'size' of undefined`.

Fix: In Cribl Edge 3.4.1.

2022-04-14 – v.3.4.0-3.4.1 – Monitoring visualizations lack color indicators [CRIBL-9510]

Problem: When you hover over the Monitoring page's graphs, the expected red/yellow/green status indicators are missing.

Fix: In Cribl Edge 3.4.2.

2022-04-07 – v.3.4.0–3.4.1 – Chain Function referencing a Pack triggers errors on Pack's Functions [CRIBL-9235]

Problem: After you define a Chain Function that references a Pack, Functions on Pipelines within that Pack will throw errors of the form: `Failed to load. Function <Function-name> is missing. Please fix, disable, or remove the Function.` You might also find that you cannot add more Functions within the Pack. (However, data might continue to flow, despite these errors.)

Workaround: Remove the Chain Function, or its Pack reference. Refactor your flow logic to avoid this combination.

Fix: In Cribl Edge 3.4.2.

2022-04-05 – v.3.4.0 through Current – Upgrade to 3.4.0 disables Worker/Node UI access [CRIBL-9175]

Problem: Upon upgrading to v.3.4.0, even if the (prior, global) Worker/Node **UI access** option was set to `Yes`, the new per–Fleet toggles are all set to `No` by default.

Workaround: On the **Manage Fleets** page, re-enable the **UI access** toggles as desired. If these toggles are not displayed (with a Free license), edit `groups.yml` to add the key-value pair `workerRemoteAccess: true` within each desired Fleet.

Fix: Version TBD.

2022-03-29 – v.3.4.0 – `this.dLogger.isSilly is not a function` errors [CRIBL-9019]

Problem: Errors of this form have been observed with multiple triggers: 1. Disabling a customized Source or Destination (it remains enabled). 2. Importing and deleting a Pack. 3. Changing a channel's logging level.

Workaround: Delete and re-create affected Sources and Destinations. (No workaround has been identified for the Pack or logging errors.)

Fix: In 3.4.1.

2022-03-23 – v.3.4.0 – Monitoring page error in distributed environments with many Edge Nodes [CRIBL-8938]

Problem: The Monitoring page displays an error where distributed environments have more than 30 Edge Nodes.

Fix: In 3.4.1.

2022-03-23 – Collect parameters values not evaluated as expressions [CRIBL-8932]

Problem: In a REST Collector's **Collect parameters** table, entering a parameter name like "earliest" as plain text will cause it to be interpreted as a literal string. This will not be evaluated as the `earliest` time-range parameter.

Workaround: Backtick the parameter name, e.g.: `` `earliest` ``.

Fix: Tooltip clarified in Cribl Edge 4.0.

2022-03-23 – Upgrading v.3.3.1 Leader to v.3.4.0 blocks upgrade to Workers [CRIBL-8918]

Problem: When you explicitly upgrade a Leader Node to 3.4.0, you can't push upgrades to Workers.

Workaround: Automatic upgrades work as expected. At **Settings** > **Global Settings** > **System > Upgrade**, set **Disable Automatic upgrades** to `No`.

Fix: In 3.4.1.

2022-03-17 – All versions through 3.4.0 – Failed systemd restarts due to erroneous `ExecStopPost` command [CRIBL-8807]

Problem: On systemd, the Cribl service can stop with an unsuccessful return code. This was caused by a malformed `ExecStopPost` command in Cribl's provided `cribl.service` file.

Workaround: In `cribl.service`, delete the whole `ExecStopPost=...` line, and change the `Restart` parameter's value from `on-failure` to `always`. (The first deletion necessitates the second change.)

Fix: Versions 3.4.1 and later install a `cribl.service` file incorporating both the above changes. **If you are upgrading from v.3.4.0 or earlier, and you notice dead Workers, check and update your existing `cribl.service` configuration to match the changes above.**

2022-03-17 – v.3.4.0 – Source PQ re-engages while draining, causing blockage/backpressure [CRIBL-8800]

Problem: A Source with Persistent Queueing enabled can mistakenly re-engage queueing while still draining its queue – leading to blocked throughput and backpressure. This occurs after a Destination goes offline and back online, once or more, leaving the Source in an undetermined queue state.

Workaround: Wait 60 seconds for another queue drain event, to see if PQ behavior goes back to normal. If the problem persists, restart Cribl Stream.

Fix: In Cribl Edge 3.4.1.

2022-03-17 – v.3.4.0 – Monitoring pages break without git [CRIBL-8799]

Problem: Unless git is installed locally, two Monitoring pages fail to display, with errors of the form `Versioning not supported`. The pages are **Monitoring** > **Overview** and **Monitoring** > **System** > **Licensing**. To resolve the errors, git must be installed even on single-instance deployments.

Workaround: Install git.

Fix: In 3.4.1.

2022-03-16 – v.3.2.2-3.4.0 – Upgrading from earlier versions degrades performance [CRIBL-8784]

Problem: After upgrading to any of the indicated versions, customers with active Splunk Destinations noticed that CPU load increased, triggering backpressure more readily.

Workaround: Skip these versions, or downgrade to a known well-behaved version.

Fix: In 3.4.1.

2022-03-11 – v.3.4.0 – In Free versions, Worker/Node UI access can't be accessed via UI [CRIBL-8707, CRIBL-8945]

Problem: In v.3.4, Cribl moved the Worker/Node **UI access** toggle from global **Settings** > **System** > **Distributed Settings** to per–Fleet toggles on the **Manage Fleets** page. But with a Free license, these controls aren't displayed in the UI at all.

Workaround: Directly edit `groups.yml` to add the key-value pair `workerRemoteAccess: true` within each desired Fleet.

Fix: In 3.4.1.

2022-03-08 – v.3.3.1 through Current – GitOps + License expiration = Catch-22 [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.

Fix: Version TBD.

2022-03-07 – v.3.2.x-3.4.x – Undercounted `total.in_bytes` dimension metrics [CRIBL-8572]

Problem: The `cribl.logstream.total.in_bytes` dimension sent to Destinations can show a lower data volume (against license quota) than `cribl.logstream.index.in_bytes`. This latter `index.in_bytes` dimension displays the correct license usage, and matches what's reported on the **Monitoring** dashboard.

Workaround: Within the `cribl_metrics_rollup` Pipeline that ships with Cribl Stream, disable the Rollup Metrics Function.

Fix: Couldn't reproduce the error.

2022-03-03 – v.3.3.x – Elasticsearch API Source (distributed mode) stops receiving data after upgrade to 3.3.x [CRIBL-8515]

Problem: After upgrading a distributed deployment to version 3.3.x, the ElasticSearch API Source might stop receiving data. This occurs when your existing Source config's **Authentication type** was set to `Auth Token`. The root cause is a 3.3.x schema change that unset this auth type to `None`.

Workaround: In the Elasticsearch API Source configuration, reset the **Authentication type** to `Auth Token`. Then commit and deploy the restored config.

Fix: In Cribl Stream 3.4.

2022-02-24 – v.3.3.x – After upgrade to 3.3.0, ElasticSearch Destination can't write to indexes whose name includes - [CRIBL-8451]

Problem: After upgrade to version 3.3.x, LogStream cannot send data to Elasticsearch indexes whose name includes a hyphen (-).

Workaround: In the Elasticsearch Destinations's **Index or Data Stream** field, surround these index names in "quotes" or backticks.

Fix: LogStream 3.3.1 added format validation and error-checking.

2022-02-18 – v.3.3.x – Worker/Leader communications blocked by untrusted TLS certificate after upgrade to 3.3.0 [CRIBL-8361]

Problem: With TLS enabled on Worker/Leader communications, after an upgrade from LogStream 3.2.x to v.3.3.x, Workers might fail to communicate with the Leader due to an untrusted SSL certificate on the Leader. Logs will show `connection errors` of the form: `"unable to verify the first certificate"` or `self signed certificate in certificate chain`.

Workaround: On each affected Worker: In the `instance.yml` file's' `distributed: master: tls:` section, set: `"rejectUnauthorized: false"`. Then restart Cribl Stream.

Fix: In 3.4.1.

2022-02-18 – v.3.3.x – Can't upgrade Workers to 3.3.x via UI [CRIBL-8346]

Problem: Attempting to upgrade on-prem Workers to version 3.3.x via the UI can fail, with an error of the form: `Group <group-name> is a managed group`.

Workaround: Edit the Leader's `$CRIBL_HOME/local/cribl/groups.yml` file to delete all instances of the key-value pair `onPrem: false`. Then, resave the file and reload the Leader.

Fix: In Cribl Stream 3.4.

2022-02-17 – v.3.2.2-3.3.x – REST Collector pagination fails on duplicate/nested attribute names [CRIBL-8352]

Problem: The REST Collector does not consistently differentiate between multiple (nested) attributes that share the same name. This can break pagination that relies on a `Response Body Attribute`.

Workaround: If possible, select a different Pagination method, or specify a unique attribute name.

Fix: In Cribl Stream 3.4.

2022-02-17 – v.3.3.0 – Do not configure InfluxDB Destination on LogStream 3.3.0 [CRIBL-8354]

Problem: Configuring InfluxDB Destinations can fail on LogStream 3.3.0. This problem has been observed only on distributed deployments, and data does flow to InfluxDB. But the InfluxDB config will not appear in LogStream's UI, and the broken config will block editing of other Destinations.

Workaround: If you need to send data to InfluxDB, skip v.3.3.0, and wait for the next release. If you encounter the broken UI, edit `outputs.yml` to remove any `influxdb_output` sections, and then restart LogStream. This will unblock UI-based editing of other Destinations.

Fix: In LogStream 3.3.1.

2022-02-10 – All versions, 3.2.1+ – Syslog Source's previewed data doesn't proceed through Routes [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.

2022-02-10 – v.3.3.0-3.5.x – Slow-Mo Notification Notifications [CRIBL-8205]

Problem: After you create a Notification on a Destination, the new Notification might take up to several minutes to appear on the Destination config modal's **Notifications** tab.

Details: This delay has not been reproduced consistently. Some Notifications appear immediately.

Fix: In Cribl Edge 4.0.

2022-02-08 – v.3.3.x – Missing event from Datadog Agent v.7.33.0+ [CRIBL-8132]

Problem: If ingesting metrics from Datadog Agent 7.33.0 or later (i.e., you have set the `DD_DD_URL` environment variable or the `dd_url` YAML config key), LogStream's Datadog Agent Source will not ingest `agent_metadata` events. This is due to a breaking change in Datadog Agent 7.33.0, which split out part of the prior `/intake/` endpoint into a new `/api/v1/metadata` endpoint.

Workaround: Use Datadog Agent v.7.32.4 or earlier.

Fix: In Cribl Stream 3.4.

2022-02-08 – All versions through 3.3.x – Syslog and Metrics Sources' default filter expression needs correction [CRIBL-8115]

Problem: In the Syslog and Metrics Sources's **Logs** tab, the auto-generated filter expression (`channel =='input:in_syslog'`) is not specific enough and yields no results, because it doesn't address these Sources' two channels (`input:in_syslog:tcp` and `input:in_syslog:udp`).

Workaround: Replace this default filter expression with `channel.startsWith('input:<input_ID>')`. For example: `channel.startsWith('input:in_syslog:')`.

Fix: In Cribl Stream 3.4.

2022-02-07 – v.3.2.2-3.4.0 – Okta integration fails after upgrading from v.3.1.3 to v.3.2.2 [CRIBL-8105]

Problem: Okta (OpenID Connect) authentication on Cribl Stream fails behind a proxy, when using environment variables (`http_proxy` or `https_proxy`).

Workaround: Configure the proxy to work in transparent mode, to bypass the `http*_proxy` variables. If the proxy is required, leave the `User Info URL` empty, and Cribl Stream can obtain user details from the `JWT token`.

Fix: In 3.4.1.

2022-02-04 – v.3.2.2-3.4.0 – Mask Function slows down post-processing Pipeline [CRIBL-8043]

Problem: A post-processing Pipeline with a Mask Function can slow down drastically, showing high CPU usage on Workers.

Workaround: If practical, promote the same Mask Function to a pre-processing or processing Pipeline.

Fix: In Cribl Stream 3.4.1.

2022-2-03 – All versions though 3.4.0 – JSON Serialize Function in post-processing Pipeline won't process [CRIBL-8013]

Problem: A JSON Serialize Function in a post-processing Pipeline will throw errors of the form: `Failed to process event in Function: Maximum call stack size exceeded.`

Workaround: Promote the Serialize Function to a processing Pipeline, upstream from the Destination.

Fix: In 3.4.1.

2022-02-02 – v.3.0.0 through Current – Upgrading a Pack overwrites Lookup and sample-data customizations [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.

Fix: Version TBD.

2022-01-11 – v.3.2.2 – Git Collapse Actions disaggregates [CRIBL-7638]

Problem: With **Git Settings** > **Collapsed actions** enabled, the combined **Commit & Push** button appears in the modal as expected, but then saving a new change splits it back into two separate buttons: **Commit** and **Deploy**.

Workaround: Disable the **Collapse actions** setting, then commit and deploy separately.

Fix: In LogStream 3.3.

2022-01-19 – v.3.2.2 – Usernames containing certain letters cause some API requests to fail [CRIBL-7728]

Problem: Usernames containing certain letters can cause some API requests to fail with an `Invalid character in header content ["cribl-user"]` error. This can happen even when the username is valid for an Identity Provider.

Workaround: In your usernames, make sure that the letters you use are limited to the letters in Latin alphabet no. 1 as defined in the ISO/IEC 8859-1 standard.

Fix: In LogStream 3.3.

2022-01-18 – v.3.2.0-3.4.0 – Diagnostics > System Info page omits some entries [CRIBL-7731]

Problem: In current Cribl Stream versions, the **Diagnostics > System Info** page/tab omits certain statistics (`sysctl`, `ulimits`, `network stats`, etc.) that were previously displayed below the `cpus` and `nets` elements.

Workaround: The hidden information is nevertheless available within diagnostic bundles.

Fix: In Cribl Stream 3.4.1.

2022-01-04 – v.3.2.x – Enabling GitOps deletes LogStream's `bin`, `logs`, and `pid` subdirectories [CRIBL-7513]

Problem: Enabling GitOps with LogStream 3.2.x, via either CLI or UI, can cause the deletion of the `bin`, `logs`, and `pid` subdirectories. This disables LogStream. The root cause is that `git` versions 2.13.1 and lower did not fully respect paths listed in `.gitignore`.

Workaround: Upgrade your `git` client to v.2.13.2 or higher, to correct the underlying behavior.

Fix: In LogStream 3.3.

2021-12-21 – v.3.2.2-4.0.x – Chain Function degrades CPU load and throughput [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.

Fix: In Cribl Edge 4.1.

2021-12-15 – v.3.2.1-3.2.2 – Backpressure when writing to S3 and other file-based Destinations [CRIBL-7364]

Problem: On file-based Destinations, enabling the **Remove staging dirs** toggle can trigger a race condition when inbound events per second reach a high rate. This leads to backpressure.

Workaround: Disable LogStream's native **Remove staging dirs** option. Instead, as the user running LogStream, set up a cron job like this:

```
crontab -e
0 1 * * * find <stagingdir> -type d -empty -mtime +1 -delete
```

Fix: In LogStream 3.3.

2021-12-13 – v.2.4.4-3.1.1 – Upgrades via UI delete sample files [CRIBL-7347]

Problem: Using the UI to upgrade LogStream versions prior to 3.1.2 will silently delete sample data files created by users. This affects using a Leader's UI (any version) to upgrade Workers running LogStream version 3.1.1 or earlier. It also affects using the UI to upgrade the Leader itself, or a Single-Instance deployment, from v.3.1.1 or earlier to any newer version.

Workarounds: 1. Upgrade pre-3.1.2 versions via the filesystem. 2. If you choose to upgrade pre-3.1.2 versions via the UI, first save to the filesystem any custom sample files that you want to preserve. (Or, to copy directly from the filesystem, LogStream stores sample files internally at `$CRIBL_HOME/data/samples`, and stores an index of all sample files in `/$CRIBL_HOME/local/cribl/samples.yml`.)

Fix: Does not affect Workers running Cribl Stream (LogStream) 3.1.2 or higher. Does not affect self-upgrades of Leaders or Single Instances running v.3.1.2 or higher.

2021-12-10 – All Versions – API Reference is temporarily unstyled

Problem: Our documentation's API Reference has temporarily lost some visual styling, while we swap in a new rendering component.

Workaround: Manually expand accordions, as needed.

Fix: Published as of 12/20/2021.

2021-12-09 – v.3.2.1 – File-based Destinations display an unavailable Persistent Queue option [CRIBL-7293]

Problem: File-based Destinations' **Backpressure behavior** drop-down misleadingly displays a `Persistent Queue` option, which is not really available on these Destinations. (Applies to Filesystem and some Amazon, Azure, and Google Cloud Destinations.) Selecting this option displays an error, and the configuration cannot be saved.

Workaround: Don't fall for clicking that fake `Persistent Queue` option.

Fix: In LogStream 3.2.2.

2021-12-07 – v.3.2.1 – System Metrics Source's documentation doesn't open in Help drawer

Problem: After clicking the System Metrics Source's Help link, the drawer displays the error: "Unable to load docs. Please check LogStream's online documentation instead." This Source's documentation is also missing from the 3.2.1 docs PDF.

Workaround: Go to the live System Metrics docs page.

Fix: In LogStream 3.2.1, as duplicate of CRIBL-6319.

2021-12-07 – v.3.1.2-3.1.3 – Increasing CPU load over time [CRIBL-7268]

Problem: CPU load increases over time, with a slow increase in memory usage. Restarting LogStream temporarily resolves these symptoms. The root cause is needlessly collecting a high volume of full-fidelity metrics from the CriblMetrics source.

Workaround: Disable the CriblMetrics Source.

Fix: Upgrade to LogStream 3.2.1, which provides an option to disable the CriblMetrics Source's **Full Fidelity** toggle.

2021-11-24 – v.3.2.0 through Current – With processing Pipelines in QuickConnect, Preview Full doesn't show Functions' results [CRIBL-7113]

Problem: If a processing Pipeline has been inserted between a QuickConnect Source and Destination, selecting the **Pipelines** page, and then selecting **Preview Full** with a data sample, doesn't show the Pipeline's effects on the **OUT** tab. This affects only Pipelines inserted in a QuickConnect connection line. (Attaching a pre-processing Pipeline to a QuickConnect Source doesn't inhibit Preview.)

Workarounds: 1. Remove the Pipeline from QuickConnect, and re-create the same Source -> Pipeline -> Destination architecture under **Routing** > **Data Routes**. 2. Rely on **Preview Simple**.

Fix: Version TBD.

2021-11-24 – v3.2.0 – Data from Collectors and Collector-based Sources isn't reaching Routes [CRIBL-7109]

Problem: Routes are not recognizing data from Collectors and from the following Collector-based Sources: Prometheus Scraper, Office 365 Activity, Office 365 Services, and Office 365 Message Trace. This data will not flow through Routes, but **will** be sent to the default Destination(s).

Workarounds: 1. In Collectors' config modals, open the **Result Routing** tab, Disable the **Send to Routes** default, and directly specify a **Pipeline** and **Destination**. (This option is not available in Prometheus Scraper or in the Office 365 Sources.) 2. Skip v.3.2.0.

Fix: In LogStream 3.2.1.

2021-11-17 – v.3.2.0 – TLS certs that use passphrases won't decrypt private keys [CRIBL-7049]

Problem: Sources whose TLS config uses a (known good) passphrase will fail to decrypt private keys. You will see a connect error, or an error of the form: `TLS validation error, is passphrase correct?`

Workarounds: 1. Edit the Leader's or single instance's `inputs.yml` file to insert a plaintext TLS passphrase. (See paths here.) Then commit and deploy the new config (distributed mode), or reload or restart the LogStream server (single instance). 2. Use a cert and key that do not require a passphrase. 3. Skip v.3.2.0.

Fix: In LogStream 3.2.1.

2021-11-17 – v.3.2.0 – Only one Chain Function works per Pipeline [CRIBL-7044]

Problem: If you add more than one Chain Function to a Pipeline, only the first will take effect. Chain Functions lower in the stack will simply pass the data down to the next Function.

Workaround: Design your data flow to require at most one Chain Function per Pipeline.

Fix: In 3.2.1.

2021-11-17 – v.3.2.0 – Exporting a Pack to another Group requires a Leader restart [CRIBL-7043]

Problem: Exporting a Pack to a different Worker Group via the UI succeeds, but opening the Pack on the target Group fails with a `Cannot read property...undefined` error.

Workaround: To resolve the error and make the target Pack accessible, restart the Leader. To prevent the error, export and import the Pack as a file.

Fix: In LogStream 3.2.1.

2021-11-17 – v.2.1.0 through Current – Re-enabling a Function group mistakenly re-enables all its Functions [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).

Fix: Version TBD.

2021-11-16 – v.3.2.0 – QuickConnected Source goes to wrong Destination [CRIBL-7013, CRIBL-7047]

Problem: Some QuickConnect connections send data to an unintended Destination. We've observed this in single-instance deployments that include the Cribl-supplied `cribl_metrics_rollup` Pipeline, or include other Pipelines/Packs with stateful Functions like Rollup Metrics or Aggregations. Data will either flow through Routes instead of your specified QuickConnect Destination, or will continue flowing to the original QuickConnect Destination after you drag the connection to a different Destination.

Workaround: Use the **Data Routes** interface to manage the Pipeline and stateful Functions indicated above. If your QuickConnect data doesn't oblige a changed QuickConnect Destination, restart LogStream. This will stop data flow to the unintended Destination, and redirect it to the intended Destination.

Fix: In Cribl Edge 3.2.1.

2021-11-10 – v.3.2.0 – GitOps `Push` workflow displayed unsupported Revert button [CRIBL-6961]

Problem: When a GitOps `Push` workflow has placed the UI in read-only mode, the Commit UI displayed a **Revert** button, even though reverting changes was not supported. Pressing the button simply triggered an error message.

Fix: In Cribl Edge/LogStream 3.2.1.

2021-10-11 – v.2.x-3.2.1 – Collectors do not filter correctly by date/time range [CRIBL-6440]

Problem: Collectors can retrieve data from undesired time ranges, instead of from the range specified. If paths are organized by date/time, this can also cause Collectors to retrieve data from the wrong paths. The root cause is that in the Collector's Run and Schedule configuration modals, time ranges are set based on the browser's time zone, whereas LogStream's backend assumes UTC.

Workaround: Offset the **Earliest** and **Latest** date/time values based on your browser's offset from UTC.

Fix: In Cribl Edge/LogStream 3.2.2 and later, the Run and Schedule modals provide a **Range Timezone** drop-down to prevent these errors.

2021-10-06 – v.3.1.2 – Monitoring page omits Collector Sources' data [CRIBL-6412]

Problem: With functioning Office 365 Activity and Office 365 Services Sources, the Job Inspector reports data being retrieved, and **Monitoring** > **Data** > **Destinations** reports data being sent out. However, **Monitoring** > **Data** > **Sources** falsely reports no data being received. The problem is isolated to this **Sources** page. It might also affect the Office 365 Message Trace and Prometheus Scraper Sources.

Workaround: Use the Source modal's **Live Data** tab, the **Monitoring** > **System** > **Job Inspector** page, and/or the **Monitoring** > **Data** > **Destinations** page to monitor throughput.

Fix: In LogStream 3.1.3.

2021-10-02 – v.3.2.x through Current – Kafka with Kerberos causes Workers' continuous restart [CRIBL-6674]

Problem: Configuring the Kafka Source or Destination with Kerberos authentication can send LogStream Workers into a continuous loop of restarts.

Workaround (multi-step):
1. In `krb5.conf`, set `dns_lookup_realm = false` and `dns_lookup_kdc = false`.
2. From `krb5.conf`'s `[realms]` section, extract the realm name (from the element name) and the FQDN (from the `kdc` key's value, stripping any port number).
3. Run `nslookup` on either or both of the realm name and the FQDN. E.g.:
`nslookup mysubdomain.confluent.io` and `nslookup kdc.kerberos-123.local`.
4. Add at least one of the returned IP addresses (which might be identical) to the `etc/hosts` file, as values in the format your platform expects.

Fix: Version TBD.

2021-09-30 – v.3.1.2 – High CPU load with CriblMetrics Source [CRIBL-6319]

Problem: Enabling the CriblMetrics Source causes high event count and high CPU load.

Workaround: Adjust the `cribl_metrics_rollup` pre-processing Pipeline to roll up a wider **Time Window** of metrics.

Fix: Upgrade to LogStream 3.2.1, which provides an option to disable the CriblMetrics Source's **Full Fidelity** toggle.

2021-09-29 – v.3.0.2-3.1.2 – Memory leak with multiple Collectors [CRIBL-6310]

Problem: Configuring multiple Collectors can lead to gradual but cumulative memory leaks. Due to a caching error, the memory can be recovered only by restarting LogStream.

Workaround: Restart LogStream on the affected Worker Nodes.

Fix: In LogStream 3.1.3.

2021-09-21 – v.3.1.1 – Azure Blob Storage Source/Destination authentication error on upgrade from v.3.0.4 [CRIBL-6236]

Problem: Upon upgrading the Azure Blob Source and/or Destination from v.3.0.4 to v.3.1.1, you might receive an error of the form: `Unable to extract accountName with provided information.`

Workaround: Change the key, and then reset it back to your desired connection string.

Fix: Not observed as of Cribl LogStream 3.1.1.

2021-09-15 – v.3.0.0–3.1.3 – High CPU usage with Google Cloud Pub/Sub Source [CRIBL-6182]

Problem: The Google Cloud Pub/Sub Source substantially increases CPU usage, which stays high even after data stops flowing. This causes throughput degradation, and more-frequent `failed to acknowledge` errors.

Workaround: Configure the Google Cloud Pub/Sub Source's **Advanced Settings** > **Max backlog** to `1000`.

Fix: In 3.2.0, which defaults the **Max backlog** to `1000`, and also relaxes the retry interval from 10 seconds to 30 seconds.

2021-09-14 – v.3.1.1 – Modifying Collector > Preview > Capture settings can break Capture elsewhere [CRIBL-6166]

Problem: Modifying the capture settings in a Collector's Run > Preview > Capture modal can improperly modify the Filter expression in Capture modals for other Collectors, Sources, and Routes/Pipelines.

Fix: In LogStream 3.1.2.

2021-09-10 – v.3.1.1 – Worker Group certificate name drop-down shows certificates from the Leader [CRIBL-6142]

Problem: When configuring certificates at **Groups** > `<group-name>` > **Settings** > **API Server Settings** > **TLS**, certificates configured on the Leader incorrectly appear on the **Certificate name** drop-down.

Fix: In LogStream 3.1.2.

2021-09-10 – v.3.1.1 – Git Collapsed Actions broken in 3.1.1 [CRIBL-6134]

Problem: With **Collapsed Actions** enabled, clicking the **Commit & Push** button has no effect. (The **Commit & Deploy** button works properly.)

Workaround: Disable **Collapsed Actions**, to restore separate **Commit** and **Git Push** buttons.

Fix: In LogStream 3.1.2.

2021-09-08 – All versions through v.3.1.1 – UDP support is currently IPv4-only [CRIBL-6106, CRIBL-6115]

Problem: Where Sources and Destinations connect over UDP, they currently support IPv4 only, not IPv6. This applies to Syslog, Metrics, and SNMP Trap Sources; and to Syslog, SNMP Trap, StatsD, StatsD Extended, and Graphite Destinations.

Workaround: Integrate via IPv4 if possible.

Fix: UDP Sources and Destinations gained IPv6 support in LogStream 3.1.2.

2021-09-02 – v.3.1.0-3.1.1 – Google Pub/Sub authentication via proxy environment variable fails [CRIBL-6086]

Problem: When LogStream's Google Cloud Pub/Sub Source and Destination attempt authentication through a proxy, using the `https_proxy` environment variable, they send an HTTP request to http://www.googleapis.com:443/oauth2/v4/token. This request fails with 504/502 errors. The root cause is a mismatch in dependency libraries, whose correction has been identified, but requires broader testing.

Workaround: Configure the proxy in `transparent` mode, to avoid relying on environment variables.

Fix: In 3.1.2.

2021-08-24 – v.3.1.0 – High CPU load with LogStream version 3.1.0 [CRIBL-6039, CRIBL-6044]

Problem: LogStream 3.1.0 added code-execution safeguards that inadvertently increased CPU load, and decreased throughput, with several Functions and most expressions.

Workaround: Downgrade to version 3.0.4.

Fix: Upgrade to version 3.1.1 or later.

2021-08-18 – v.3.1.0 – Clone Collector option broken [CRIBL-5977]

Problem: Clicking a 3.1.0 Collector modal's **Clone Collector** button simply closes the modal. (If you have unsaved changes, you'll first be challenged to confirm closing the parent modal – but the expected cloned modal won't open.)

Workaround: Click **+ Add New** to re-create your original Collector's config from scratch, adding any desired modifications.

Fix: In LogStream 3.1.1.

2021-08-18 – All versions through 3.1.0 – Tabbed code blocks broken on in-app docs [CRIBL-5972]

Problem: When docs with tabbed code blocks are opened in the Help drawer, the default (leftmost) tab seizes focus. Other tabs will not display when clicked.

Workaround: Click the blue/linked page title atop the Help drawer to open the same page on docs.cribl.io, where all tabs can be selected.

Fix: In LogStream 3.1.1.

2021-08-14 – v.3.1.0 – Splunk Load Balanced Destination does not migrate auth type [CRIBL-5940]

Problem: In a Splunk Load Balanced Destination with **Indexer discovery** enabled and a corresponding **Auth token** defined, upgrading to LogStream 3.1.0 corrupts the **Auth token** field's value.

Workaround: Set the **Authentication method** to **Manual** and resave the token's value.

Fix: In 3.1.1.

2021-08-11 – v.3.1.0 – `C.Secret()` values are undefined in Collectors [CRIBL-5926]

Problem: Calling the C.Secret() internal method within a Collector field resolves incorrectly to an `undefined` substring. E.g., in URL fields, `C.Secret()` values will resolve to `/undefined/` path substrings.

Workarounds: 1. Use `C.vars` and a Global Variable, instead of using this method. 2. Root cause is that `C.Secret()` in Collectors and Pipeline Functions has access only to secrets that were created before the last restart. Therefore, restart Worker Processes to refresh the method's access.

Fix: In 3.1.1.

2021-08-10 – v.3.1.0 – Pre-processing Pipelines break Flows display [CRIBL-5909]

Problem: Attaching a pre-processing Pipeline to a Source breaks the **Monitoring** > **Flows (beta)** page's display. Attempting to remove Sources/Destinations from that page's selectors throws a cryptic `Sankey` error.

Workaround: Temporarily detach pre-processing Pipelines if you want to check Flows.

Fix: Upgrade to version 3.1.1 or later.

2021-07-29 – v.3.0.2-3.0.4 – Upgrades via UI require broader permissions [CRIBL-5774]

Problem: Upgrading from v.3.0.x via the UI requires the `cribl` user to be granted write permission on the parent directory above `$CRIBL_HOME`. The symptom is an error message of the form: `Upgrade failed: EACCES: permission denied, mkdir '/opt/unpack.xxxxxxx.tmp'`.

Workaround: Either adjust permissions, or upgrade via the filesystem. For complete instructions, see Upgrading.

Fix: Does not affect LogStream 3.1 or higher.

2021-07-26 – v.3.0.x–3.1.0 – Packs with orphaned lookups block access to Worker Groups [CRIBL-5738]

Problem: If a Pack references a lookup file that's missing from the Pack, pushing the Pack to a Worker Group will block access to the Group's UI. You will see an error message of the form: "The Config Helper service is not available because a configuration file doesn't exist... Please fix it and restart LogStream."

Workaround: On the Leader Node, review the config helper logs (`$CRIBL_HOME/log/groups/<group>/*.log`) to see which references are broken. (In a single-instance deployment, see `$CRIBL_HOME/log/*.log`.) Then manually resolve these references in the Pack's configuration.

Fix: Upgrade to version 3.1.1 or later.

2021-07-20 – v.3.0.3-3.4.0 – Can't add Functions to a Pipeline named `config` [CRIBL-5706]

Problem: You cannot add Functions to a Pipeline if the Pipeline is named `config`, because this name conflicts with the reserved route for the **Create Pipeline** dialog.

Workarounds: Don'tcha name your Pipelines `config`.

Fix: In Cribl Edge 3.4.1.

2021-07-06 – All versions through Current – Duplicate Workers/ Worker GUIDs [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.

Fix: Version TBD.

2021-07-02 – v.3.0.2–3.0.3 – Sample file's last line not displayed upon upload [CRIBL-5595]

Problem: When uploading (attaching) a sample data file, the file's final line is not displayed in the **Add Sample Data** modal.

Workarounds: This is a UI bug only. LogStream correctly processes the complete sample data, which should show up when viewing the sample afterwards (e.g., within a Pipeline's preview pane).

Fix: In LogStream 3.1.0.

2021-07-02 – All versions through 3.0.x – Date fields misleadingly preview with string symbol [CRIBL-5594]

Problem: In Preview or Capture, incoming events like `_raw` will be displayed in the right pane with an `α` symbol that indicates string data. However, calling `new Date()` and then `C.Time.strptime()` methods in an Eval Function will return `null` on the OUT tab.

Cause: Due to the nature of JSON serialization, the incoming event's `Date` field is misleadingly subsumed under the event's `α` string symbol. It's actually a structured type, not a string...yet.

Workaround: If you see unexpected `null` results, stringify the datetime field as you extract it, e.g.: `new Date().toISOString()`. Feeding the resulting field to Time methods should return datetime strings as expected.

Fix: In LogStream 3.1.0.

2021-06-22 – v.3.0.0–3.0.3 – Internal `C.Text.relativeEntropy()` method – broken typeahead and preview [CRIBL-5534]

Problem: The `C.Text.relativeEntropy()` internal method is missing from JavaScript expressions' typeahead drop-downs. You can manually type or paste in the method, and save your Function and Pipeline, but LogStream's right Preview pane will (misleadingly) always show the method returning `0`.

Workarounds: Use other means (such as the **Live** button) to preview and verify that the method is (in fact) returning valid results.

Fix: In LogStream 3.1.0.

2021-05-20 – v.3.0.0 – Multiple Functions Break LogStream 3.0 Pipelines [CRIBL-5311, CRIBL-5766]

Problem: After upgrade to LogStream 3.0.0, including any of the following Functions in a Pipeline can break the Pipeline: GeoIP, Redis, DNS Lookup, Reverse DNS, Tee. Symptom is an error of the form: `Pipeline process timeout has occurred.` Less seriously, including these Functions in a Pipeline can suppress Preview's display of fields/values.

Workarounds: If you use these Functions in your Pipelines, stay with (or restore) a pre-3.0 version until LogStream 3.0.1 is available.

Fix: CRIBL-5311 fixed in LogStream 3.0.1. CRIBL-5766 fixed in LogStream 3.2.1.

2021-05-19 – v.3.0.0 – Leader's Changes fly-out stays open after Commit

Problem: In the Leader's left nav, the Changes fly-out remains stuck open after you commit pending changes.

Workarounds: Hover or click away. Then hover or click back to reopen the fly-out.

Fix: In LogStream 3.0.1.

2021-05-18 – v.3.0.0 – Packs > Export in "Merge" mode omits schemas and custom Functions

Problem: Exporting a Pack with the export mode set to `Merge` omits schemas and custom Functions configured within the Pack's **Knowledge > Schemas**.

Workarounds: 1. Change the export mode to `Merge safe`, and export again. 2. If that doesn't preserve the schema and Functions, revert to `Merge` export mode; install the resulting Pack onto its target(s); and then manually copy/paste the schema(s) and Functions from the source Pack's UI to the target Pack's UI.

Fix: In LogStream 3.0.1.

2021-05-17 – v.3.0.0–3.2.0 – Can't Enable KMS on Worker Group after installing license

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.

Fix: Version TBD.

2021-05-10 – v.2.4.5-3.0.3 – Elasticsearch Destination, with Auto version discovery, doesn't send Authorization header

Problem: When the Elasticsearch Destination has Basic Authentication enabled, and its **Elastic version** field specifies `Auto` version discovery, LogStream fails to send the configured username and password credentials along with its API initial request. Elasticsearch responds with an HTTP 401 error.

Workaround: Explicitly set the **Elastic version** to either `7.x` or `6.x` (depending on your Elasticsearch cluster's version); then stop and restart LogStream to pick up this configuration change.

Fix: In LogStream 3.1.0.

2021-05-04 – v.2.4.5-3.0.1 – Office 365 Message Trace Source skips events

Problem: The Event Breaker Rule provided for the Office 365 Message Trace Source mistakenly presets the **Default timezone** to `ETC/GMT-0`. This setting causes LogStream to discover events but not collect them.

Workaround: Reset the Rule's **Default timezone** to `UTC`, then click **OK** and resave the Ruleset.

Fix: In 3.0.2.

2021-05-03 – v.2.4.4–3.0.1 – Rollup Function suppresses `sourcetype` metrics

Problem: `sourcetype` metrics can be suppressed when the Cribl Internal > CriblMetrics Source is enabled and the `cribl_metrics_rollup` pre-processing Pipeline is attached to a Source.

Workarounds: Disabling the pre-processing pipeline restores `sourcetype` and any other missing data. However, without the rollup, a much higher data volume will be sent to the indexing tier.

Fix: In LogStream 3.0.2.

2021-04-20 – v.2.4.3–2.4.5 – Orphaned S3 staging directories

Problem: Using the S3 Destination, defining a partitioning expression with high cardinality can proliferate a large number (up to millions) of empty directories. This is because LogStream cleans up staged files, but not staging directories.

Workaround: Programmatically or manually delete stale staging directories (e.g., those older than 30 days).

Fix: In LogStream 3.0.2.

2021-04-12 – v.2.4.4-3.0.0 – Splunk Sources do not support multiple-metric events

Problem: LogStream's Splunk Sources do not support multiple-measurement metric data points. (LogStream's Splunk Load Balanced Destination does.)

Fix: In LogStream 3.0.1.

2021-04-07 – v.2.4.2–2.4.5 – Google Cloud Storage Destination fails to upload files > 5 MB

Problem: The Google Cloud Storage Destination might fail to put objects into GCS buckets. This happens with files larger than 5 MB, and causes the Google Cloud API to report a vague `Invalid argument` error.

Workaround: Set the **Max file size (MB)** to 5 MB. Also, reduce the **Max file open time (sec)** limit from its default `300` (5 minutes) to a shorter interval, to prevent files from growing to the 5 MB threshold. (Tune this limit based on your observed rate of traffic flow through the Destination.)

Fix: In LogStream 3.0.0.

2021-03-31 – v.2.4.4-2.4.5 – Local login option visible even when disabled

Problem: The **Log in with local user** option is displayed to users even when you have disabled **Settings > Authentication > Allow local auth** for an OpenID Connect identity provider.

Workaround: Advise users to ignore this button. Although visible, it will not function.

Fix: In LogStream 3.0.0.

2021-03-31 – v.2.4.0–2.4.4 – Splunk TCP and LB Destinations' Workers trigger OOM errors and restart

Problem: With a Splunk TCP or Splunk Load Balanced Destination created after upgrading to LogStream 2.4.x, Workers' memory consumption may grow without bound, leading to out-of-memory errors. The API Process will restart the Workers, but there might be temporary outages.

Workaround: Toggle the Destination's **Advanced Settings > Minimize in-flight data loss** slider to `No`. This will preserve Processes killed by OOM conditions.

Fix: In LogStream 2.4.5.

2021-03-31 – v.2.4.4-2.4.5 – OpenID Connect authentication always shows local-auth fallback

Problem: Even if OpenID Connect external authentication is configured to disable **Allow local auth**, LogStream's login page displays a **Log in with local user** button.

Workaround: Do not click that button.

Fix: In LogStream 3.0.0.

2021-03-31 – v.2.4.4 – Authentication options mistakenly display Cribl Cloud

Problem: The **Settings > Authentication > Type** drop-down offers a `Cribl Cloud` option, which is not currently functional. Attempting to configure and save this option could lock the `admin` user out of LogStream.

Workaround: Do not select, configure, or save that option.

Fix: In LogStream 2.4.5.

2021-03-30 – v.2.4.4 – Can't disable some Sources from within their config modals

Problem: In configuration modals for the Azure Blob Storage and Office 365 Message Trace Sources, the **Enabled** slider cannot be toggled off, and its tooltip doesn't appear.

Workaround: Disable your configured Source (where required) from the **Manage Blob Storage Sources** or the **Manage Message Trace Sources** page.

Fix: In LogStream 2.4.5.

2021-03-29 – v.2.4.x – SpaceOut Destination is broken

Problem: Within the SpaceOut game, you cannot shoot, and your player is immortal.

Workaround: There are other video games. After we defeat COVID, you'll even be able to buy a PS5.

Fix: Restored in LogStream 2.4.5.

2021-03-24 – v.2.4.x – Cribl App for Splunk blocks admin password changes, configuration changes, and Splunk-based authentication

Problem: Attempting to change the admin password via the UI triggers a 403/Forbidden message. You can reset the password by editing `users.json`, but can't save configuration changes to Settings, Pipelines, etc., because RBAC Roles are not properly applied.

Workaround: Using a 2.3.x version of the App enables **local** authentication and enables changes to Cribl/LogStream passwords and configuration/settings.

Fix: In LogStream 2.4.4.

2021-03-22 – v.1.7.0-2.4.3 – Azure Event Hubs Destination: Compression must be manually disabled

Problem: LogStream's Azure Event Hubs Destination provides a **Compression** option that defaults to `Gzip`. However, compressed Kafka messages are not yet supported on Azure Event Hubs.

Workaround: Manually reset **Compression** to `None`, then resave Azure Event Hubs Destinations.

Fix: In LogStream 2.4.4.

2021-03-17 – v.2.4.2-2.4.3 – Parser Function > List of Fields copy/paste fails

Problem: When copying/pasting **List of Fields** contents between Parser Functions via the Copy button, the paste operation inserts unintended metadata instead of the original field references.

Workaround: Manually re-enter the second Parser Function's **List of Fields**.

Fix: In LogStream 2.4.4.

2021-03-13 – v.2.4.3 – UI can't find valid TLS .key files, blocking Master restarts and Worker reconfiguration

Problem: After upgrading to v.2.4.3, the UI fails to recognize valid TLS `.key` files, displaying spurious error messages of the form: "File does not exist: `$CRIBL_HOME/local/cribl/auth/certs/<keyname>key`." An affected Master will not restart. Affected Workers will restart, but will not apply changes made through the UI.

Workaround: Ideally, specify an absolute path to each key file, rather than relying on environment variables. If you're locked out of the UI, you'll need to manually edit the referenced paths within these configuration files in LogStream subdirectories: `local/cribl/cribl.yml` (General > API Server TLS settings) and/or `local/_system/instance.yml` (Distributed > TLS settings). Contact Cribl Support if you need assistance. A more drastic workaround is to disable TLS for the affected connections.

Fix: In LogStream 2.4.4.

2021-03-12 – v.2.4.2-2.4.5 – Redis Function with specific username can't authenticate against Redis 6.x ACLs

Problem: The Redis Function, when used with a specific username and Redis 6.x's Access Control List feature, fails due to authentication problems.

Workaround: In the Function's **Redis URL** field, point to the Redis `default` account, either with a password (e.g., `redis://default:Password1@192.168.1.20:6379`) or with no password (redis://192.168.1.20:6379). Do not specify a user other than `default`.

Fix: In LogStream 3.0.

2021-03-09 – v.2.4.3 – Splunk Destinations' in-app docs mismatch UI's current field order

Problem: For the Splunk Single Instance and Splunk Load Balanced Destinations, the in-app documentation omits the UI's **Advanced Settings** section. Some fields are documented out-of-sequence, or are omitted.

Workaround: Refer to the UI's tooltips, to the corrected Splunk Single Instance and Splunk Load Balanced online docs, and/or to the corrected PDF.

Fix: In LogStream 2.4.4.

2021-03-08 – v.2.4.3 – Enabling Git Collapse Actions breaks Commit & Deploy

Problem: After enabling **Settings > Distributed Settings > Git Settings > General > Collapse Actions**, selecting **Commit & Deploy** throws a 500 error.

Workaround: Disable the **Collapse Actions** setting, then commit and deploy separately.

Fix: In LogStream 2.4.4.

2021-03-08 – v.2.4.3 – S3 Collector lacks options to reuse HTTP connections and allow-self signed certs

Problem: As of v.2.4.3, LogStream's AWS-related Sources & Destinations provide options to reuse HTTP connections, and to establish TLS connections to servers with self-signed certificates. However, the S3 Collector does not yet provide these options.

Fix: In LogStream 2.4.4.

2021-03-04 – v.2.4.2 – Esc key closes both Event Breaker Ruleset modals

Problem: After adding a rule to a **Knowledge > Event Breaker Ruleset**, pressing `Esc` closes the parent Ruleset modal along with the child Rule modal.

Workaround: Close the Rule modal by clicking either its **Cancel** button or its close box.

Fix: In LogStream 2.4.3.

2021-03-04 – v.2.4.2 – Aggregations Function in post-processing Pipeline addresses wrong Destination

Problem: An Aggregations Function, when used in a post-processing Pipeline, sends data to LogStream's Default Destination rather than to the Pipeline's attached Destination.

Workaround: If applicable, use the Function in a processing or pre-processing Pipeline instead.

Fix: In LogStream 2.4.3.

2021-02-25 – v.2.4.2 – On Safari, Event Breaker shows no OUT events

Problem: When viewing an Event Breaker's results on Safari, no events are displayed on the Preview pane's **OUT** tab.

Workaround: Use another supported browser.

Fix: In LogStream 2.4.3.

2021-02-22 – v.2.4.3 – Collection jobs UI errors

Problem: Collection jobs are missing from the **Monitoring > Sources** page, even though they are returned by metric queries. Also, the **Job Inspector > Live** modal displays an empty, unintended **Configure** tab.

Workaround: Use the Job Inspector to access collection results. Ignore the **Configure** tab.

Fix: In LogStream 2.4.4.

2021-02-19 – v.2.4.2 – Upon upgrade, Git remote repo setting breaks, blocking Worker Groups

Problem: If a Git remote repo was previously configured, upgrading to LogStream v.2.4.2 throws errors of this form upon startup: `Failed to initialize git repository. Config versioning will not be available...Invalid URL...`. The Master cannot commit or deploy to any Worker Group.

Workarounds: 1. Downgrade back to v.2.4.1 (or your previous working version). 2. Switch from Basic authentication to SSH authentication against the repo, to remove the username from requests.
(The apparent root cause is Basic/http auth using a valid URL and username, but missing a password.)

Fix: In LogStream 2.4.3.

2021-02-19 – v.2.4.0-2.4.2 – Splunk (S2S) Forwarder access control blocks upon upgrade to LogStream 2.4.x

Problem: If Splunk indexers have forwarder tokens enabled, and worked with LogStream 2.3.x before, upgrading to LogStream 2.4.x causes data to stop flowing.

Workaround: If you encounter this problem, rolling back to your previously installed LogStream version (such as v.2.3.4) resolves it.

Fix: In LogStream 2.4.3.

2021-02-10 – v.2.4.0-2.4.1 – With Splunk HEC Source, JSON payloads containing embedded objects trigger high CPU usage

Problem: Splunk HEC JSON payloads containing nested objects trigger high CPU usage, due to a flaw in JSON parsing.

Workaround: If you encounter this problem, rolling back to your previously installed LogStream version (such as v.2.3.4) resolves it.

Fix: In LogStream 2.4.2.

2021-01-30 – v.2.4.0 – Worker Nodes cannot connect to Master

Problem: Worker Nodes cannot connect to the Master after the Master is upgraded to v.2.4.0.

Workaround: Disable compression on the Workers. You can do so through the Workers' UI at **System Settings > Distributed Settings > Master Settings > Compression**, or by commenting out this line in each Worker's `cribl.yml` config file:

```
compression: gzip
```

Fix: In LogStream 2.4.1.

2021-01-25 – v.2.4.0 – S3 collection stops working due to auth secret key issues.

Problem: S3 collection stops after upgrade to 2.4.0 due to secret key re-encryption.

Workaround: Re-configure S3, save and re-deploy.

Fix: In LogStream 2.4.1.

2021-01-14 – v.2.4.0 – Google Cloud Storage Destination Needs Extra Endpoint to Initialize

Problem: The Google Cloud Storage Destination fails to initialize, displaying an error of the form: `Bucket does not exist!`

Workaround: In the `outputs.yml` file, under your `cribl-gcp-bucket` key endpoint, add: `https://storage.googleapis.com`. (in a single-instance deployment, locate this file at `$CRIBL_HOME/local/cribl/outputs.yml`. In a distributed deployment, locate it at `$CRIBL_HOME/groups/<group name>/local/cribl/outputs.yml`.)

Fix: In LogStream 2.4.1.

2021-01-14 – v.2.4.0 – Worker Groups' Settings > Access Management Is Absent from UI

Problem: In this release, the **Worker Groups > <group-name> > System Settings** UI did not display the expected **Access Management**, **Authentication**, and **Local Users** sections.

Workaround: Manually edit the `users.json` file.

Fix: In LogStream 2.4.1.

2021-01-13 – v.2.4.0 – Route Filters Aren't Copied to Capture Modal

Problem: On the **Routes** page, selecting **Capture New** in the right pane does not copy custom **Filter** expressions to the resulting **Capture Sample Data** modal. That modal's **Filter Expression** field always defaults to `true`.

Workarounds: 1. Bypass the **Capture New** button. Instead, from the Route's own ••• (Options) menu, select **Capture**. This initiates a capture with the **Filter Expression** correctly populated. 2. Copy/paste the expression into the **Capture Sample Data** modal's **Filter Expression** field. Or, if the expression is displayed in that field's history drop-down, retrieve it.

Fix: In LogStream 2.4.1.

2021-01-13 – v.2.4.0 – Destinations' Documentation Doesn't Render from UI

Problem: Clicking the **Help** Help link in a Destination's configuration modal displays the error message: "Unable to load docs. Please check LogStream's online documentation instead."

Workarounds: 1. Go directly to the online Destinations docs, starting here. 2. Follow the UI link to the docs landing page, click through to open or download the current PDF, and scroll to its Destinations section.

Fix: In LogStream 2.4.1.

2021-01-13 – v.2.4.0 – `Esc` Key Doesn't Consistently Close Modals

Problem: Pressing `Esc` with focus on a modal's drop-down or slider doesn't close the modal as expected. (Pressing `Esc` with focus on a free-text field, combo box, or nothing does close the modal – displaying a confirmation dialog first, if you have unsaved changes.)

Workarounds: Click the **X** close box at upper right, or click **Cancel** at lower right.

Fix: In LogStream 2.4.1.

2020-12-17 – v.2.3.0-2.4.0 – Free-License Expiration Notice, Blocked Inputs

Problem: LogStream reports an expired Free license, and blocks inputs, even though Free licenses in v.2.3.0 do not expire.

Workaround: This is caused by time-limited Free license key originally entered in a LogStream version prior to 2.3.0. Go to **Settings > Licensing**, click to select and expand your expired Free license, and click **Delete license**. LogStream will recognize the new, permanent Free license, and will restore throughput.

Fix: In LogStream 2.4.1.

2020-11-14 – v.2.3.3 – Null Fields Redacted in Preview, but Still Forwarded

Problem: Where event fields have null values, LogStream (by default) displays them as struck-out in the right Preview pane. The preview is misleading, because the events are still sent to the output.

Workaround: If you do want to prevent fields with null values from reaching the output, use an Eval Function, with an appropriate Filter expression, to remove them.

Fix: Preview corrected in LogStream 2.3.4.

2020-10-27 – v.2.3.2 – Cannot Name or Save New Event Breaker Rule

Problem: After clicking **Add Rule** in a new or existing Event Breaker Ruleset, the **Event Breaker Rule** modal's **Rule Name** field is disabled. Because **Rule Name** is mandatory field, this also disables saving the Rule via the **OK** button.

Fix: In LogStream 2.3.3.

2020-10-12 – v.2.3.1 – Deleting One Function Deletes Others in Same Group

Problem: After inserting a new Function into a group and saving the Pipeline, deleting the Function also deletes other Functions lower down in the same group.

Fix: In LogStream 2.3.2.

Workaround: Move the target Function out of the group, resave the Pipeline, and only then delete the Function.

2020-09-27 – v.2.3.1 – Enabling Boot Start as Different User Fails

Problem: When a root user tries to enable boot-start as a different user (e.g., using `/opt/cribl/bin/cribl boot-start enable -u <some-username>`), they receive an error of this form:

```
error: found user=0 as owner for path=/opt/cribl, expected uid=NaN.
Please make sure CRIBL_HOME and its contents are owned by the uid=NaN by running:
[sudo] chown -R NaN:[$group] /opt/cribl
```

Fix: In LogStream 2.3.2.

Workaround: Install LogStream 2.2.3 (which you can download here), then upgrade to 2.3.1.

2020-09-17 – v.2.3.0 – Worker Groups menu tab hidden after upgrade to LogStream 2.3.0

Problem: Upon upgrading an earlier, licensed LogStream installation to v.2.3.0, the **Worker Groups** tab might be absent from the Master Node's top menu.

Fix: In LogStream 2.3.1.

Workaround: Click the **Home > Worker Groups** tile to access Worker Groups.

2020-09-17 – v.2.3.0 – Cannot Start LogStream 2.3.0 on RHEL 6, RHEL 7

Problem: Upon upgrading to v.2.3.0, LogStream might fail to start on RHEL 6 or 7, with an error message of the following form. This occurs when the user running LogStream doesn't match the LogStream binary's owner. LogStream 2.3.0 applies a restrictive permissions check using `id -un <uid>`, which does not work with the version of `id` that ships with these RHEL releases.

```
id: 0: No such user
ERROR: Cannot run command because user=root with uid=0 does not own executable
```

Fix: In LogStream 2.3.1.

Workaround: Update your RHEL environment's `id` version, if possible.

2020-09-17 – v.2.3.0 – Cannot Start LogStream 2.3.0 with OpenId Connect

Problem: Upon upgrading an earlier LogStream installation to v.2.3.0, OIDC users might be unable to restart the LogStream server.

Fix: In LogStream 2.3.1.

Workaround: Edit `$CRIBL_HOME/default/cribl/cribl.yml` to add the following lines to its the `auth` section:

```
filter_type: email_whitelist
scope: openid profile email
```

2020-06-11 – v.2.1.x – Can't switch from Worker to Master Mode

Problem: In a Distributed deployment, attempting to switch Distributed Settings from Worker to Master Mode blocks with a spurious "Git not available...Please install and try again" error message.

Fix: In LogStream 2.3.0.

Workaround: To initialize `git`, switch first from Worker to Single mode, and then from Single to Master mode.

2020-05-19 – v.2.1.x – Login page blocks

Problem: Entering valid credentials on the login page (e.g., `http://localhost:9000/login`) yields only a spinner.

Fix: In LogStream 2.3.0.

Workaround: Trim `/login` from the URL.

2020-02-22 – v.2.1.x – Deleting resources in `default/`

Problem: In a Distributed deployment, deleting resources in `default/` causes them to reappear on restart.

Workaround/Fix: In progress.

2019-10-22 – v.2.0.0 – In-product upgrade issue on v2.0

Problem: Using in-product upgrade feature in v.1.7 (or earlier) fails to upgrade to v.2.0, due to package-name convention change.

Workaround/Fix: Download the new version and upgrade per steps laid out here.

2019-08-27 – v.1.7 – In-product upgrade issue on v.1.7

Problem: Using in-product upgrade feature in v.1.6 (or earlier) fails to upgrade to v.1.7 due to package name convention change.

Workaround/Fix: Download the new package and upgrade per steps laid out here.

2019-03-21 – v.1.4 – S3 stagePath issue on upgrade to v.1.4+

Problem: When upgrading from v.1.2 with a S3 output configured, `stagePath` was allowed to be undefined. In v.1.4+, `stagePath` is a required field. This might causing schema violations when upgrading older configs.

Workaround/Fix: Reconfigure the output with a valid `stagePath` filesystem path.